MULTI-TENANT GEN AI DOCUMENT INTELLIGENCE SYSTEM ON AMAZON BEDROCK- 3 TIER ARCHITECTURE

PRIYANKA RAJAGOPAL

GitHub

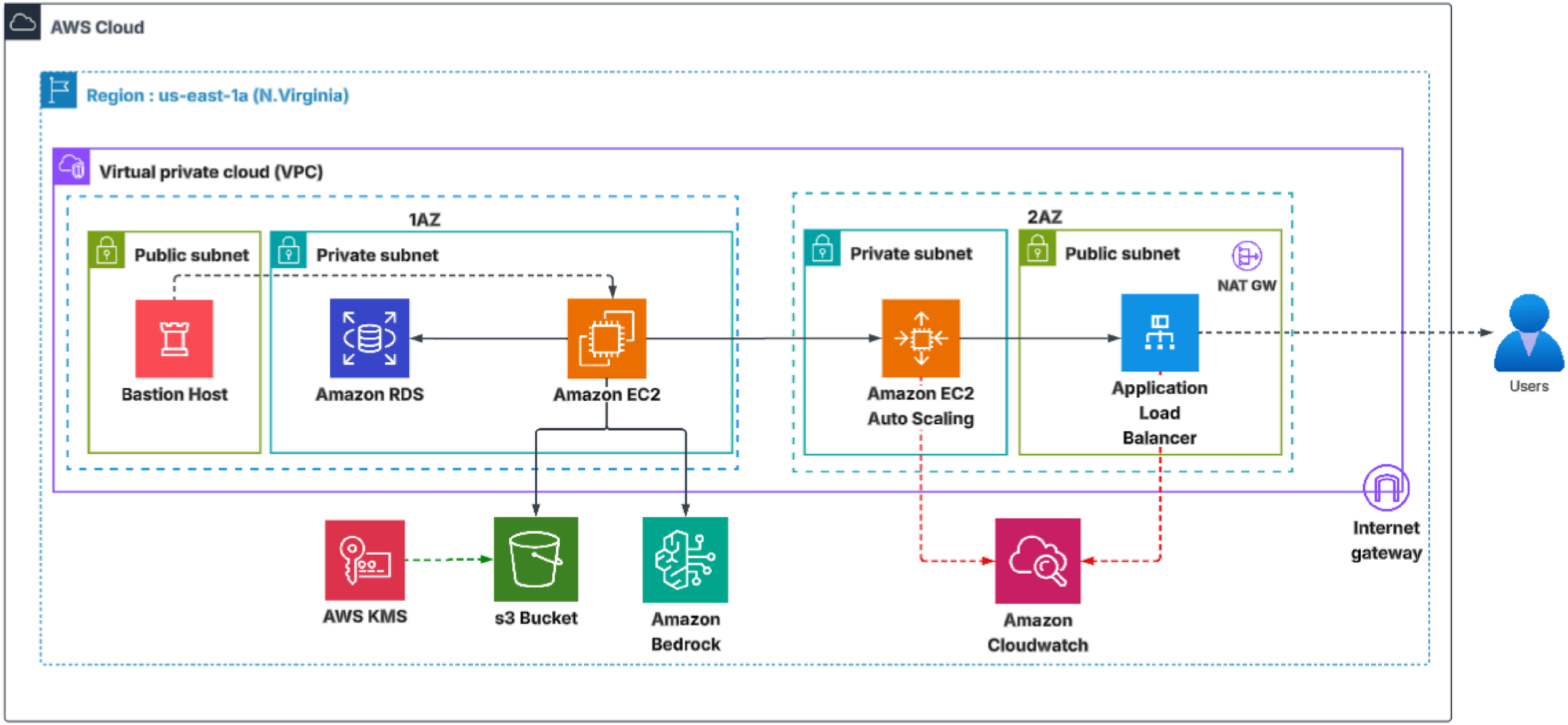
PROJECT OVERVIEW

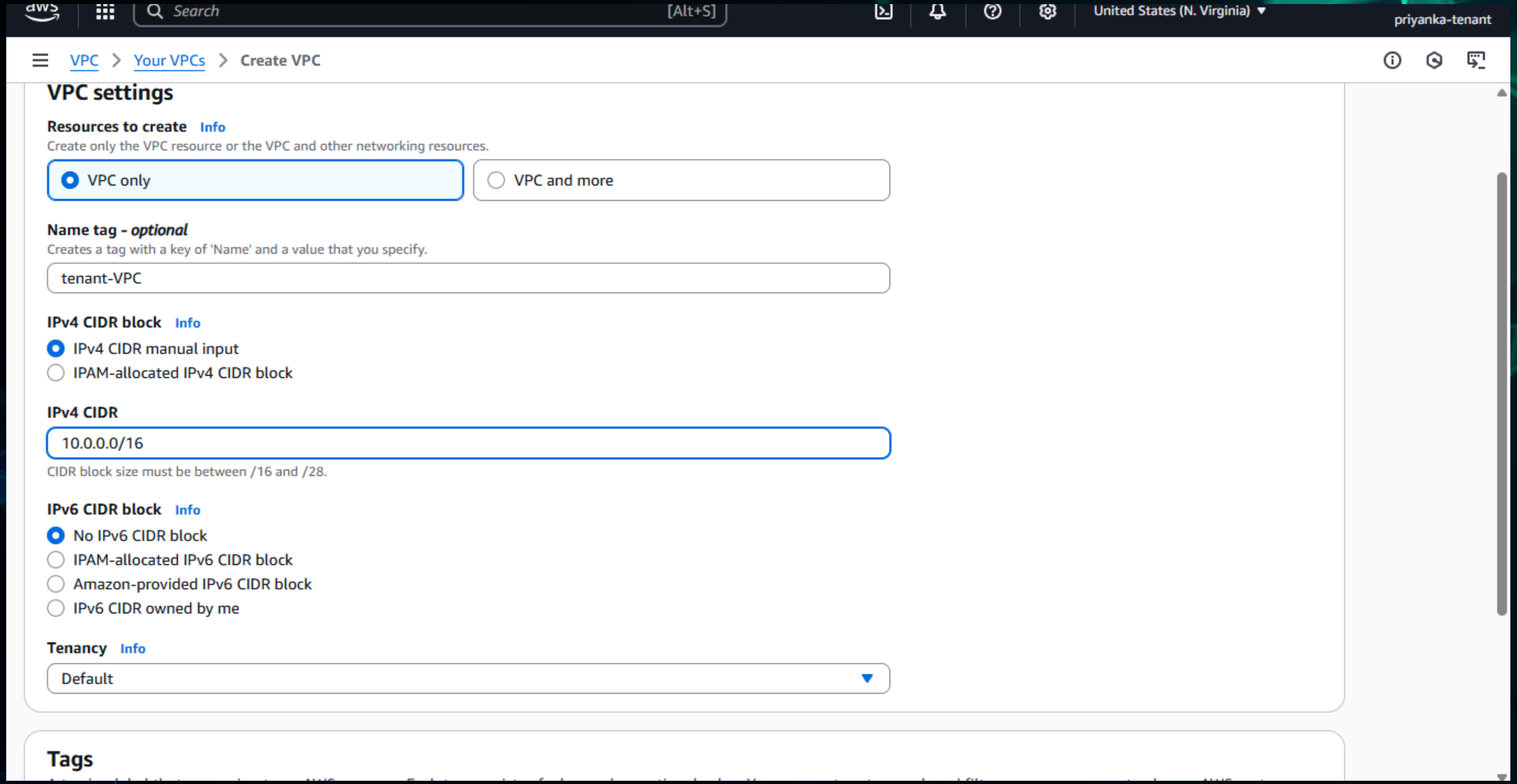
- This project presents an AI multi-tenant PDF summarization system using Amazon Bedrock, delivering secure, scalable, and automated document intelligence in the cloud. Users upload PDF documents through a public endpoint and receive concise summaries generated by Amazon Bedrock's Titan Text foundation model, demonstrating the use of AI and machine learning services.

- The solution is built entirely on AWS and follows a 3-tier, high-availability, multi-AZ architecture, with an Application Load Balancer as the presentation tier, EC2 instances in an Auto Scaling Group as the application tier, and Amazon S3 and Amazon RDS (Aurora MySQL) as the data tier. Strong security is enforced using IAM least privilege access, KMS encryption for S3, private subnet isolation, and CloudWatch for monitoring and observability.

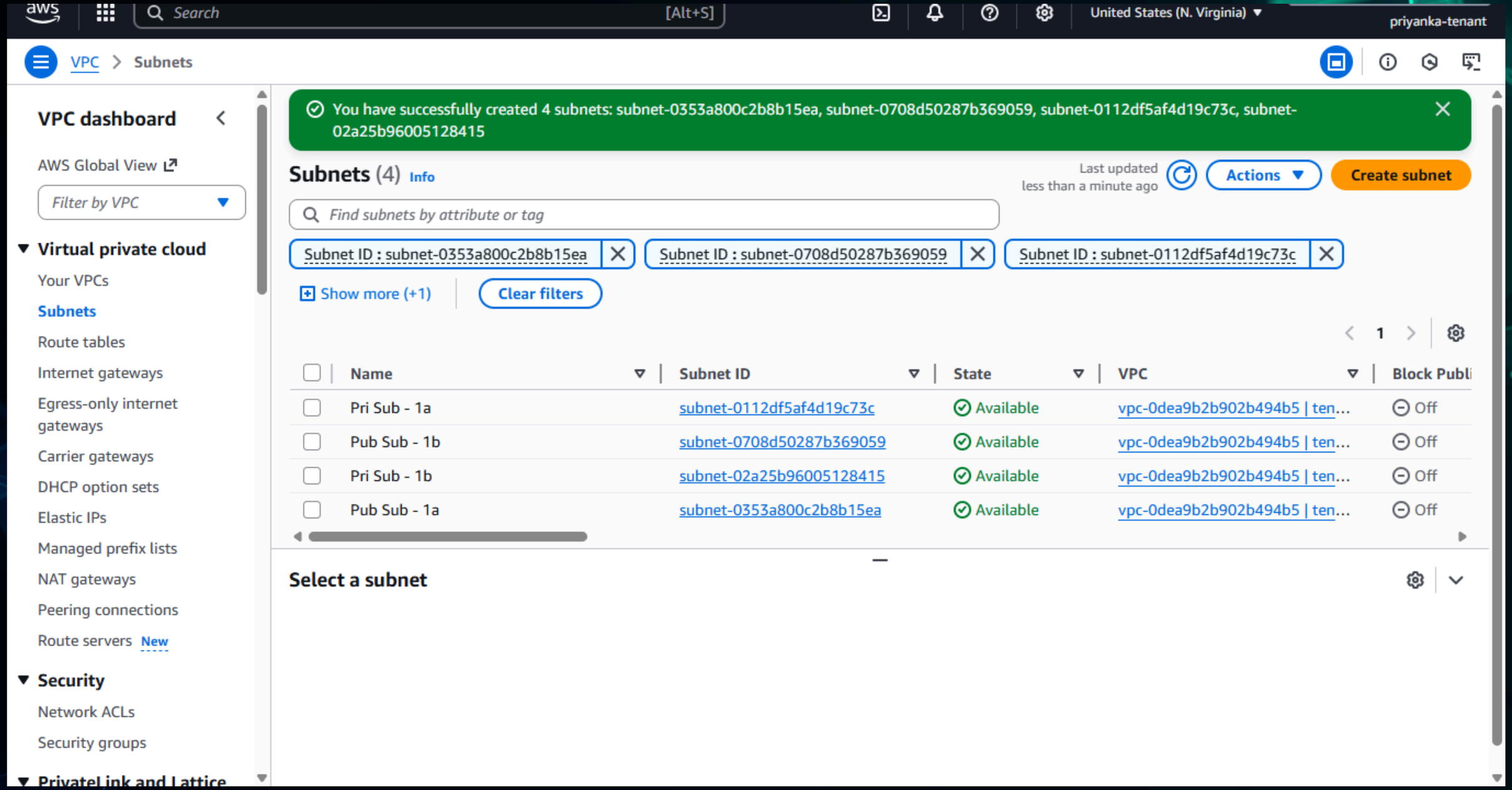
SERVICES USED

ARCHITECTURE

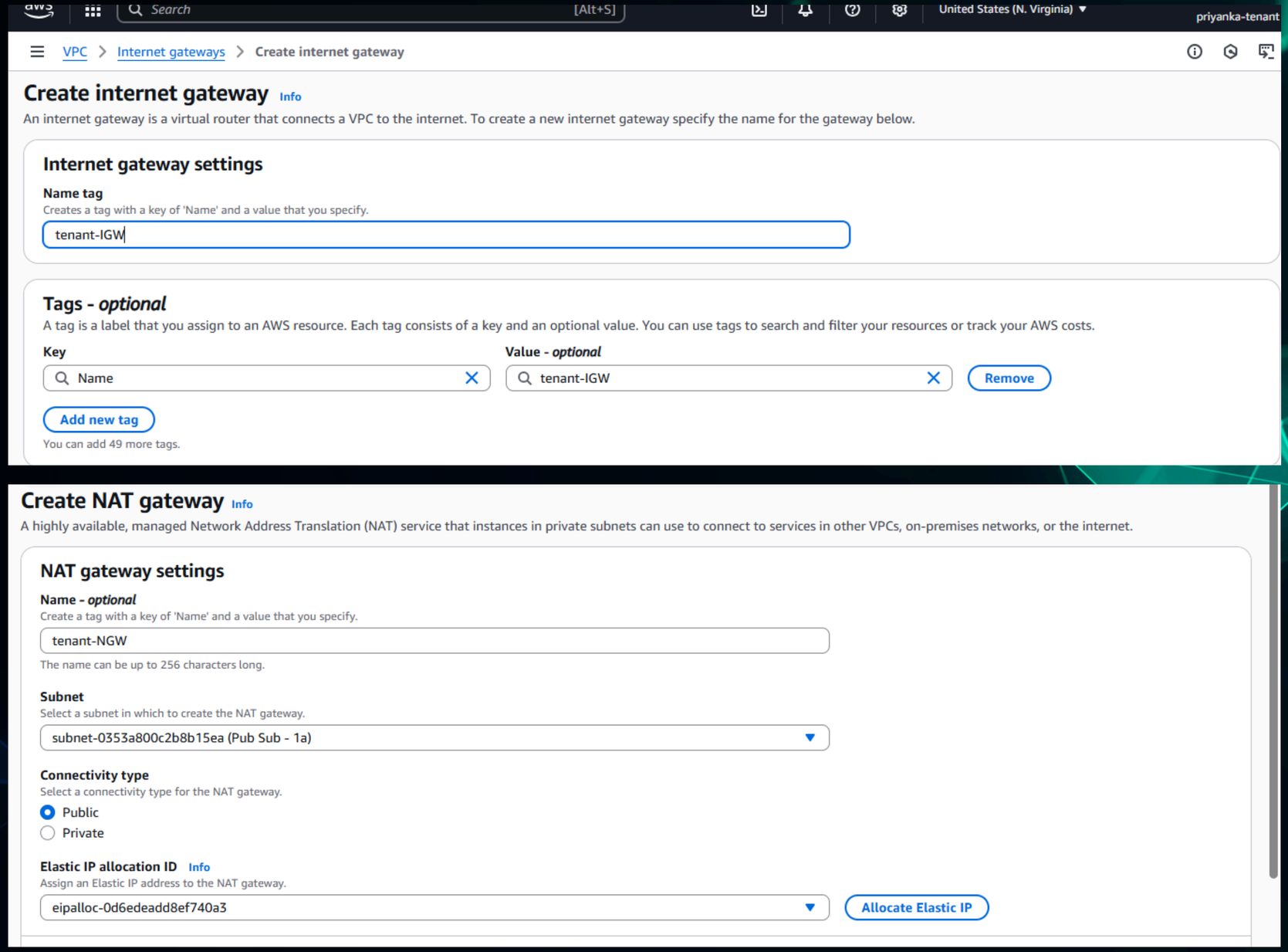




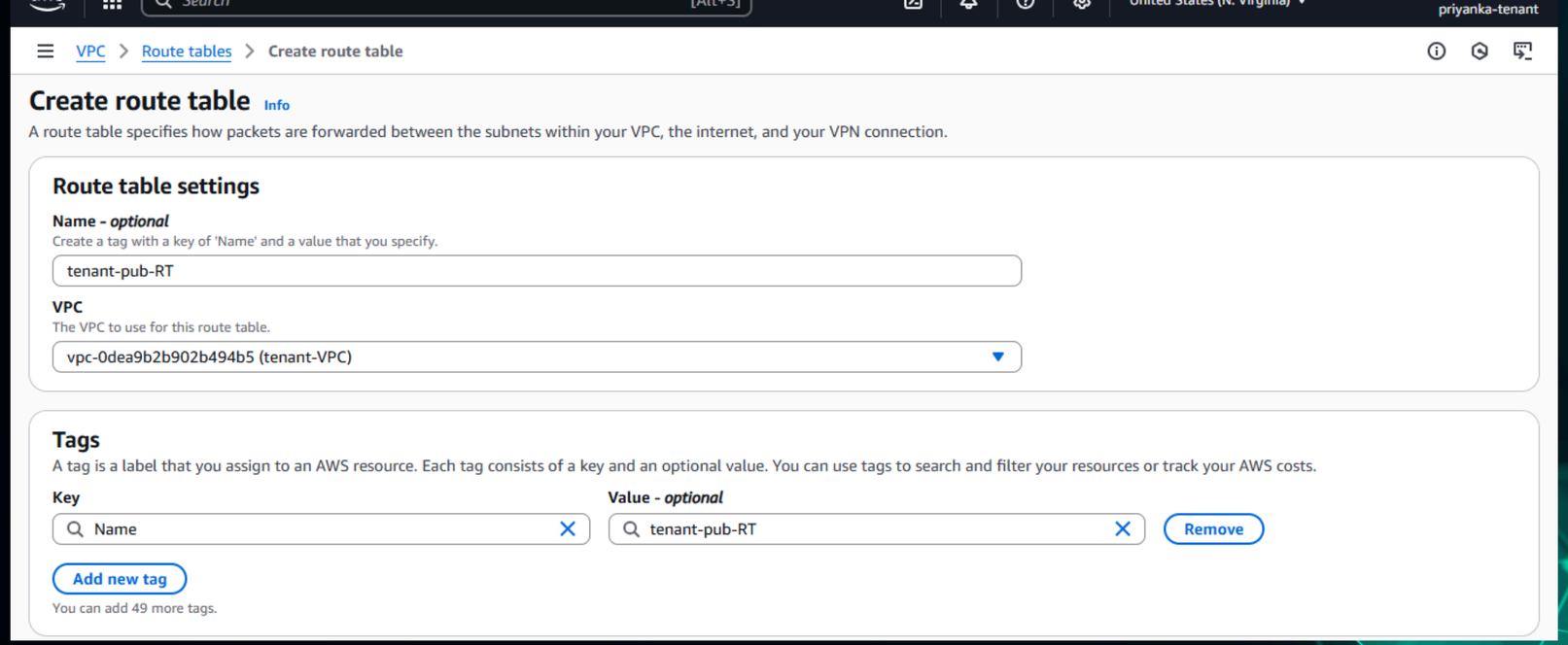
Created VPC with CIDR 10.0.0.0/16

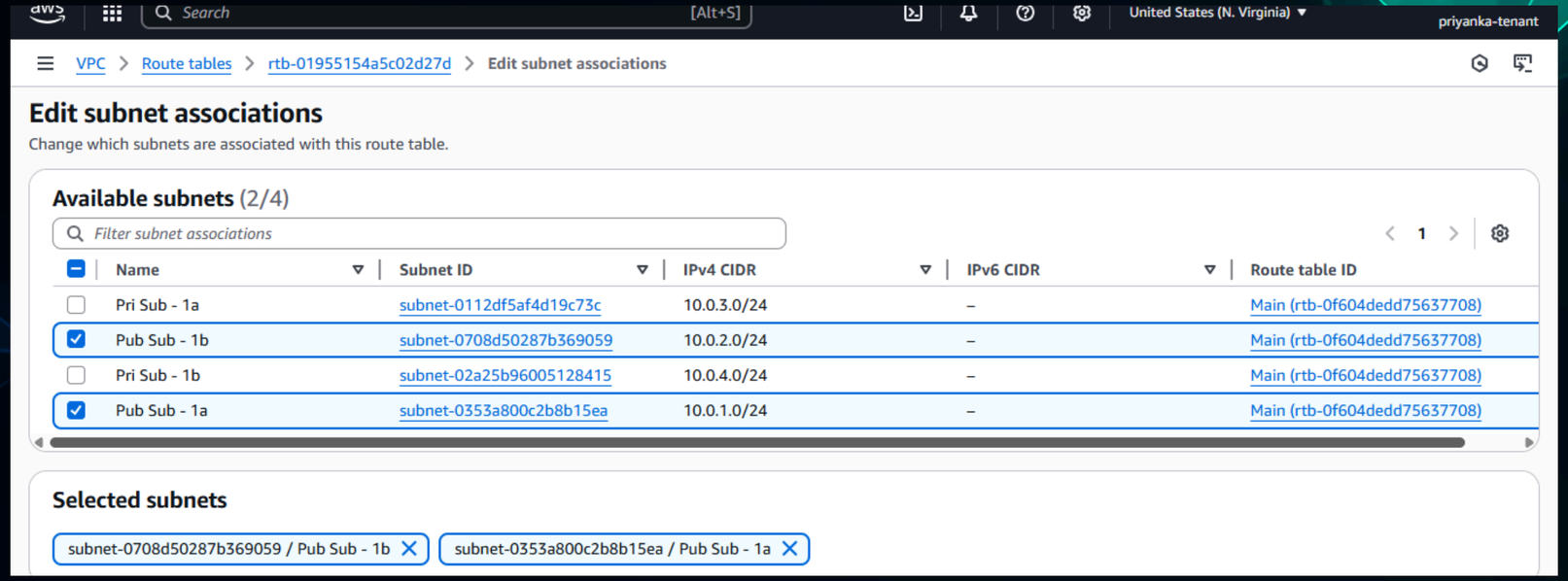


Created 4 Subnets in 2 Availability zones,2 Public Subnets and 2 Private Subnets.

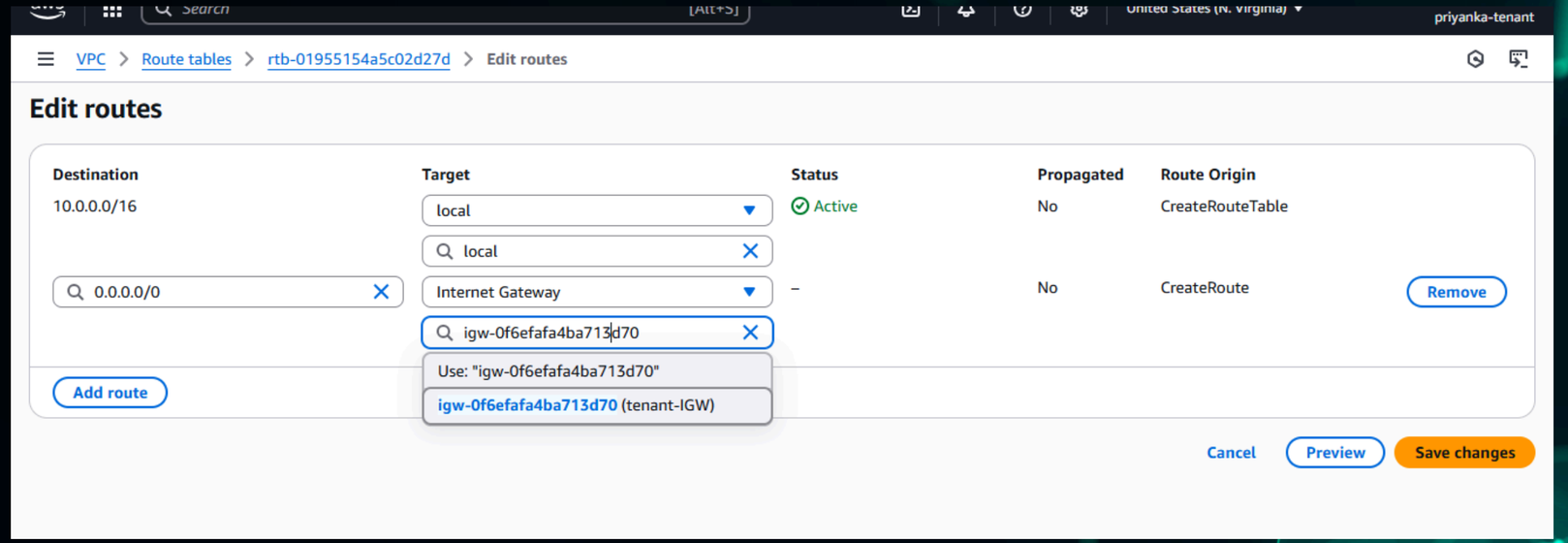


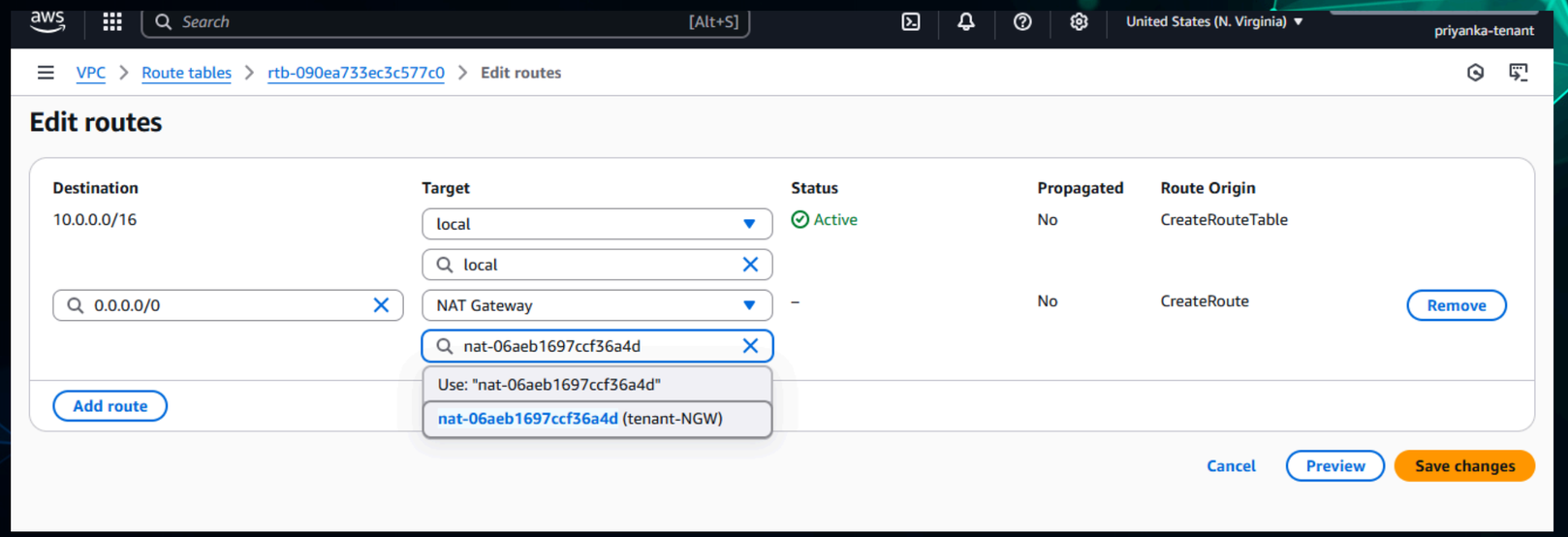
Created and Attached Internet gateway ,Allocated Elastic IP and created NAT Gateway.



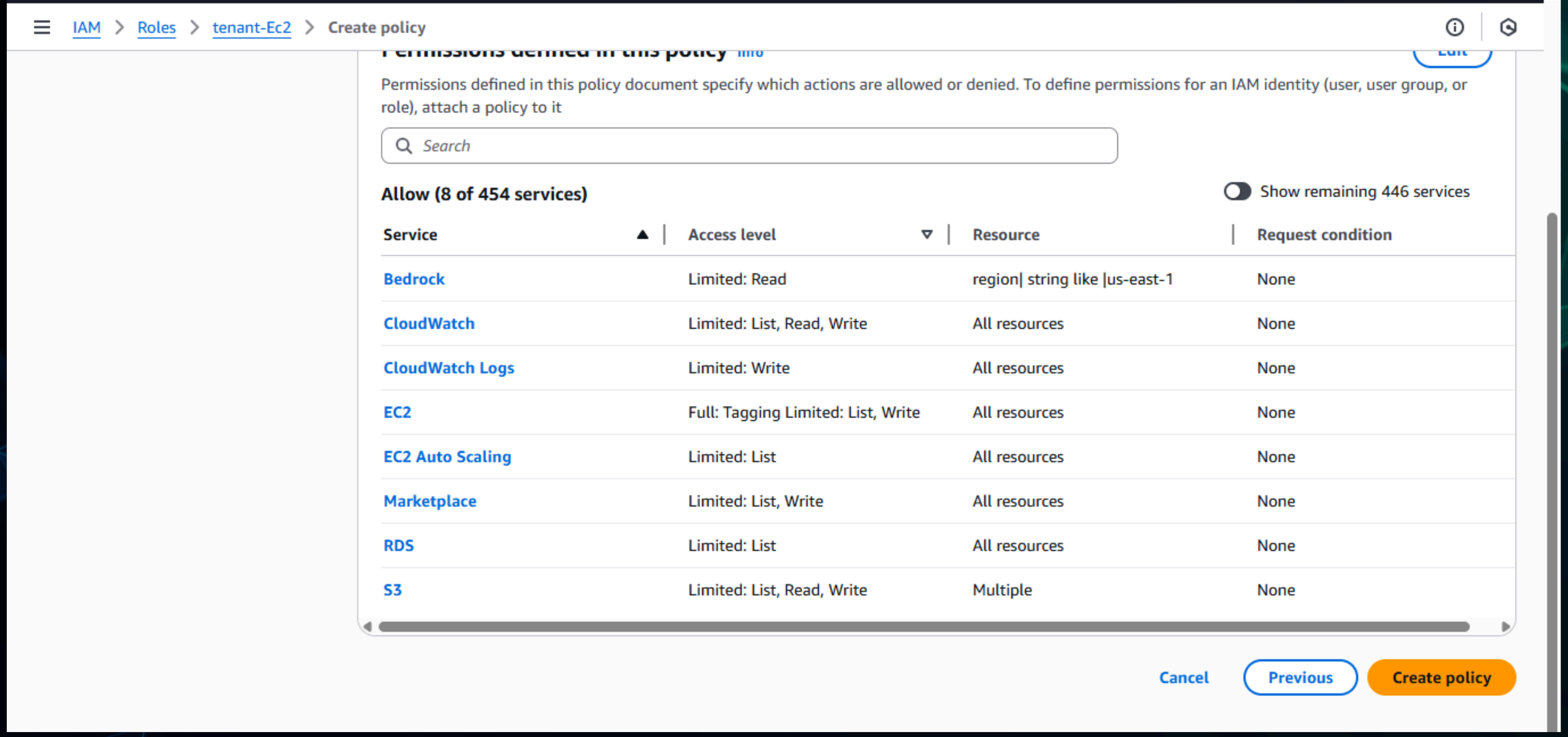


Created 2 Route tables and associated with the Public and Private Subnets.

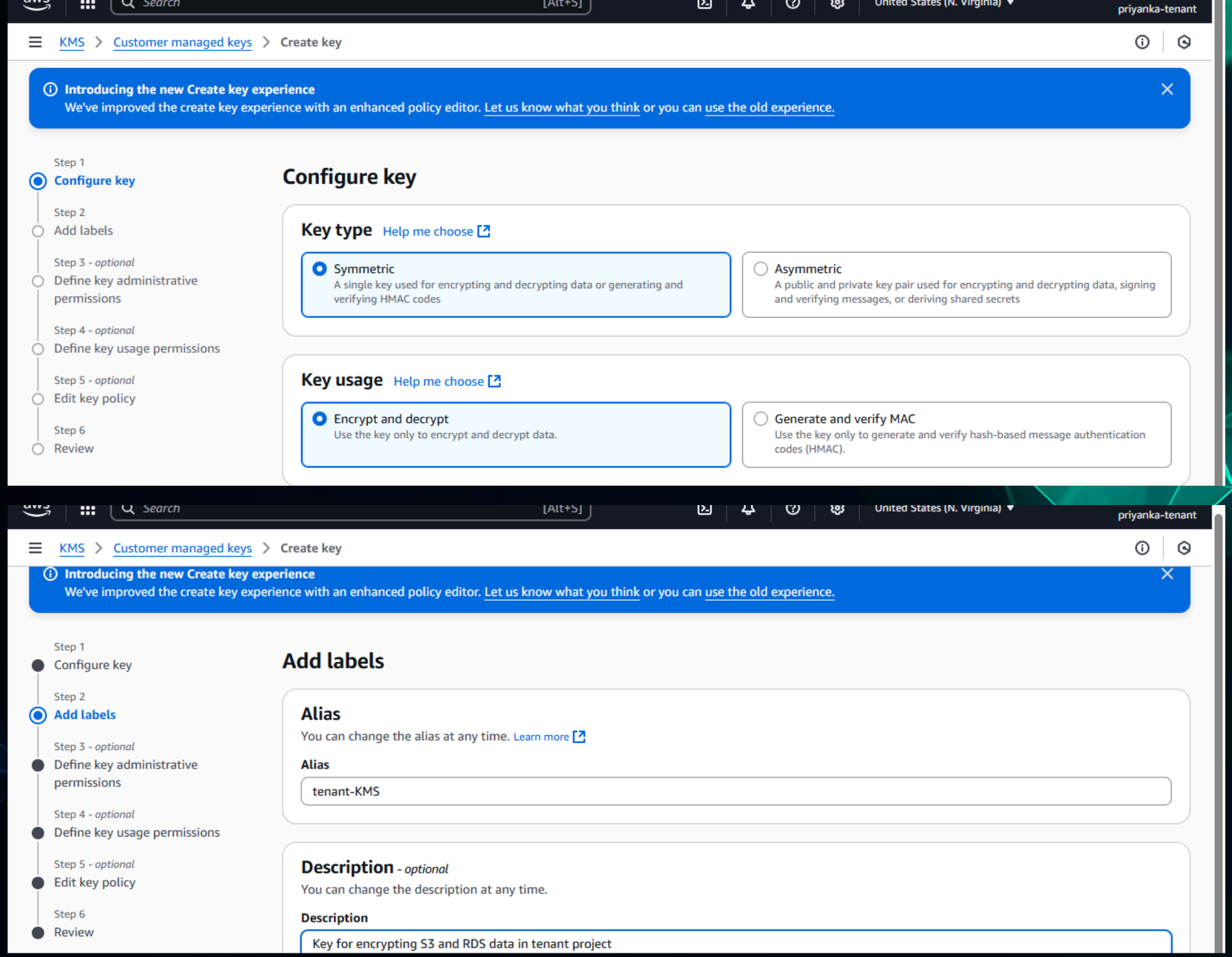




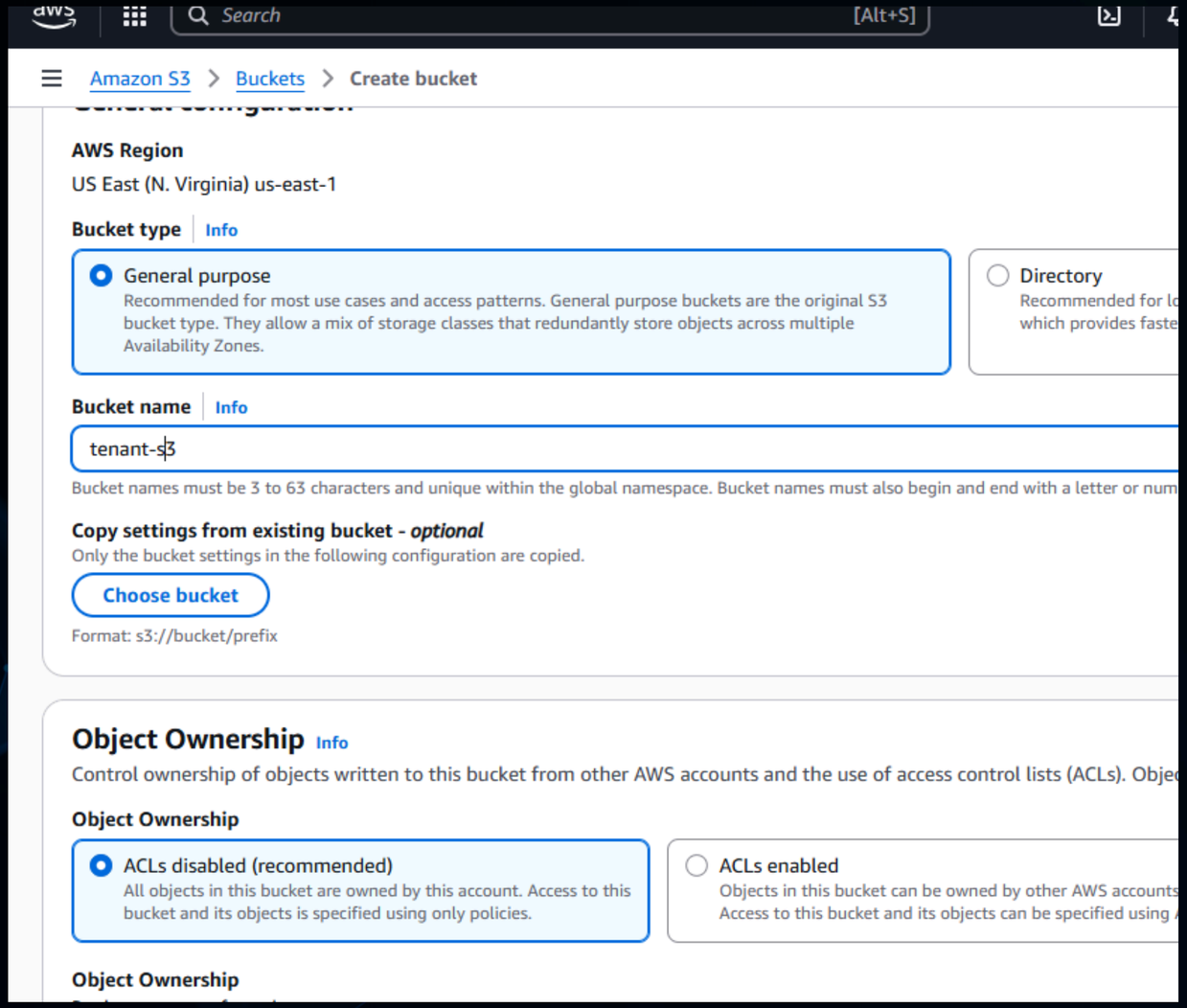
Public subnets route 0.0.0.0/0 to the Internet Gateway (IGW) to allow internet access for the ALB and Bastion Host.Private subnets route 0.0.0.0/0 to the NAT Gateway to enable secure outbound access without public exposure.

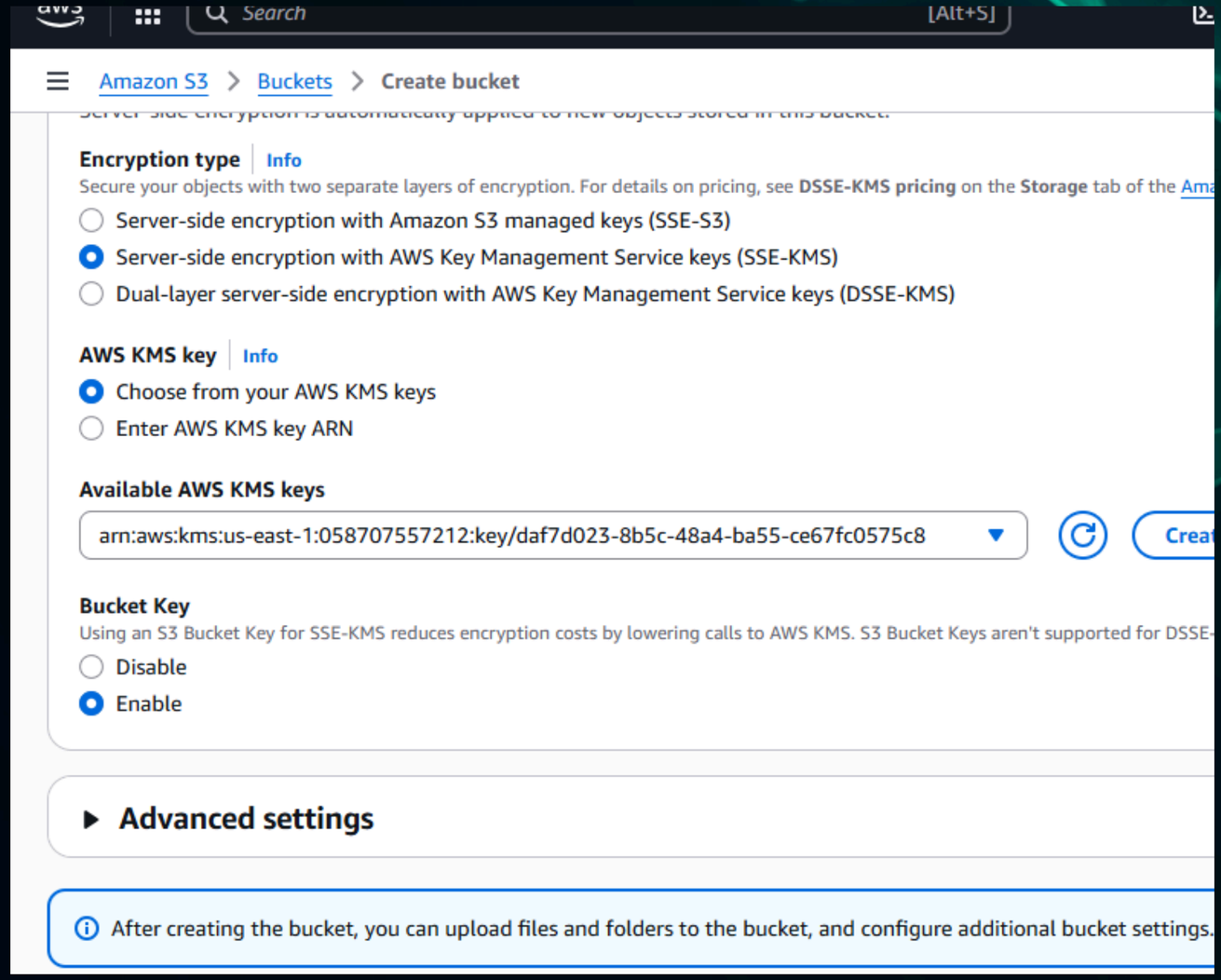


Created IAM roles for the services with the Least Privilege principle.

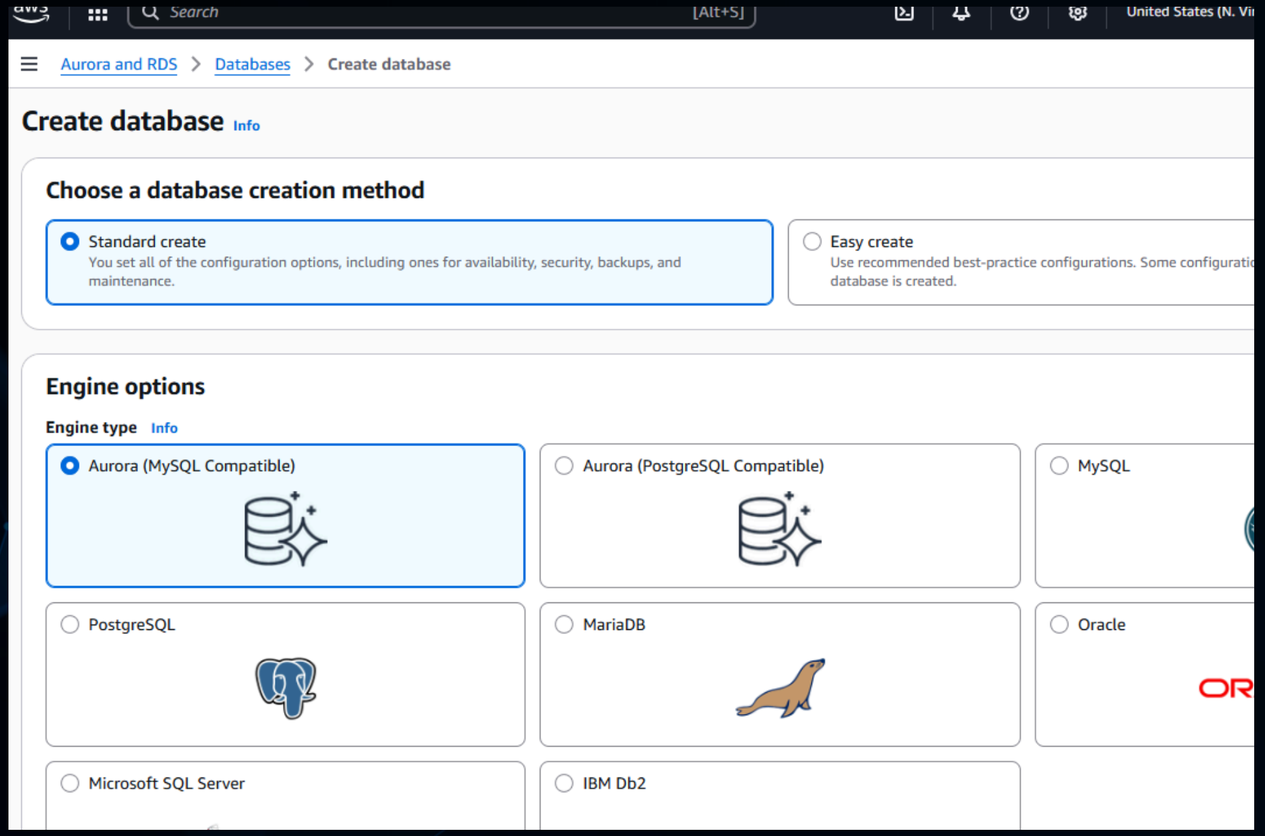


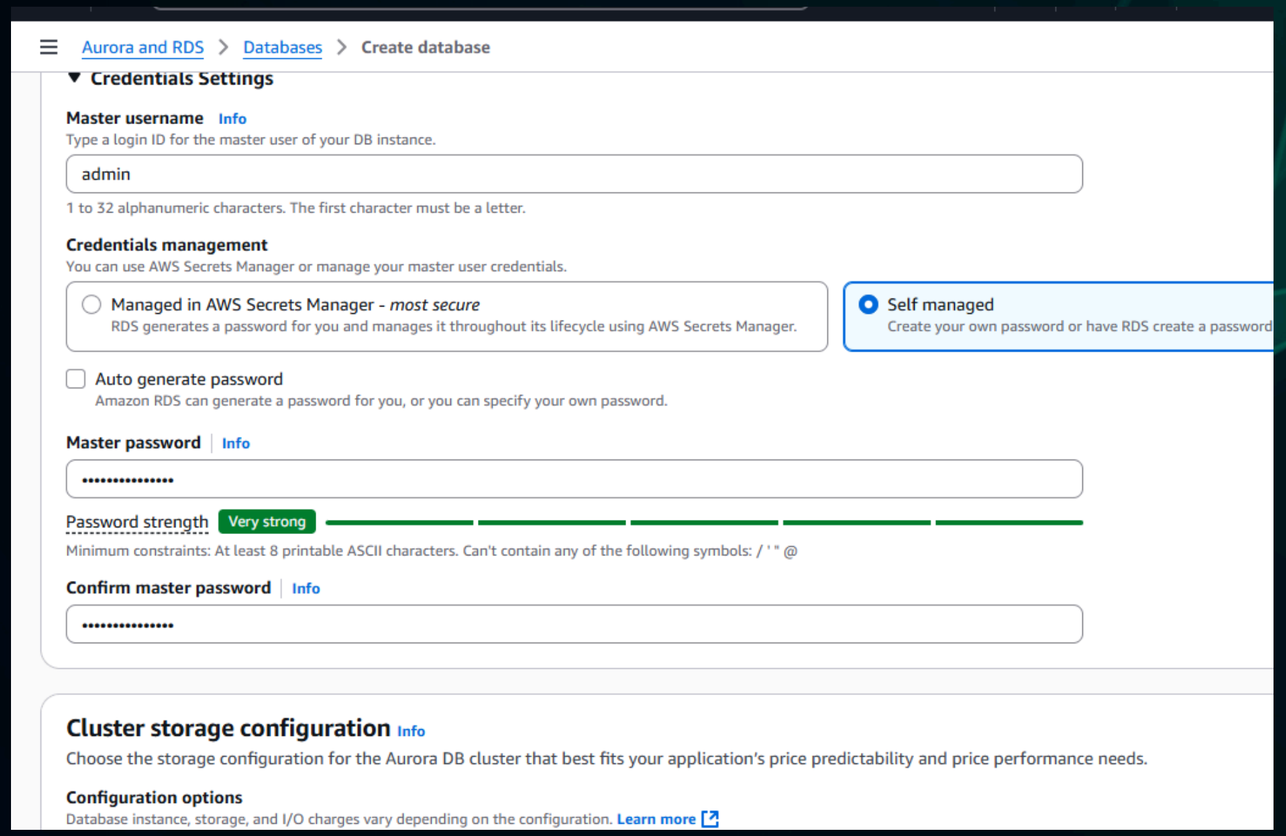
Created KMS encryption key for S3.



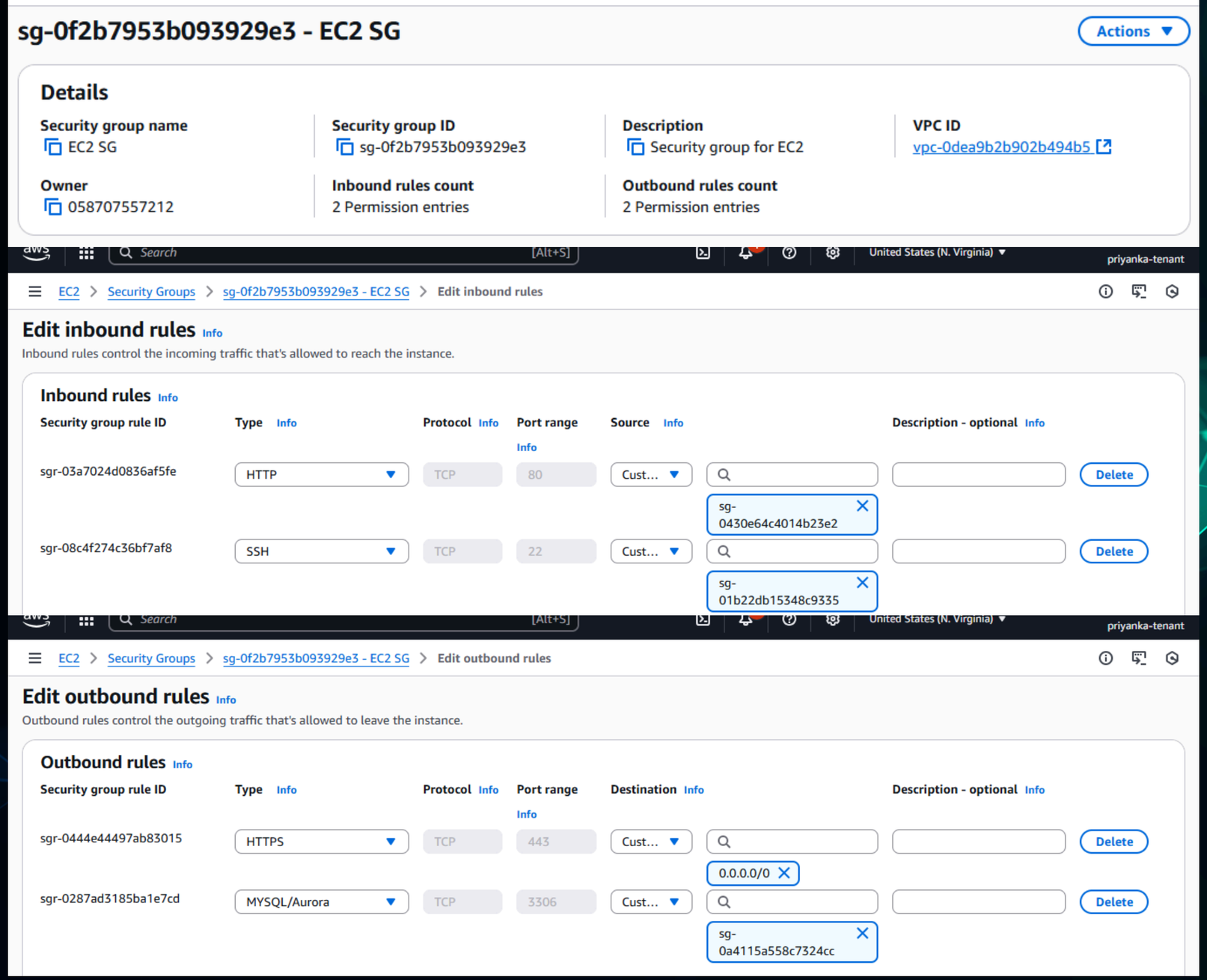


Created S3 Bucket for the Raw storage of PDF/Book when uploaded from ALB and attached the KMS encryption key SSE-KMS.

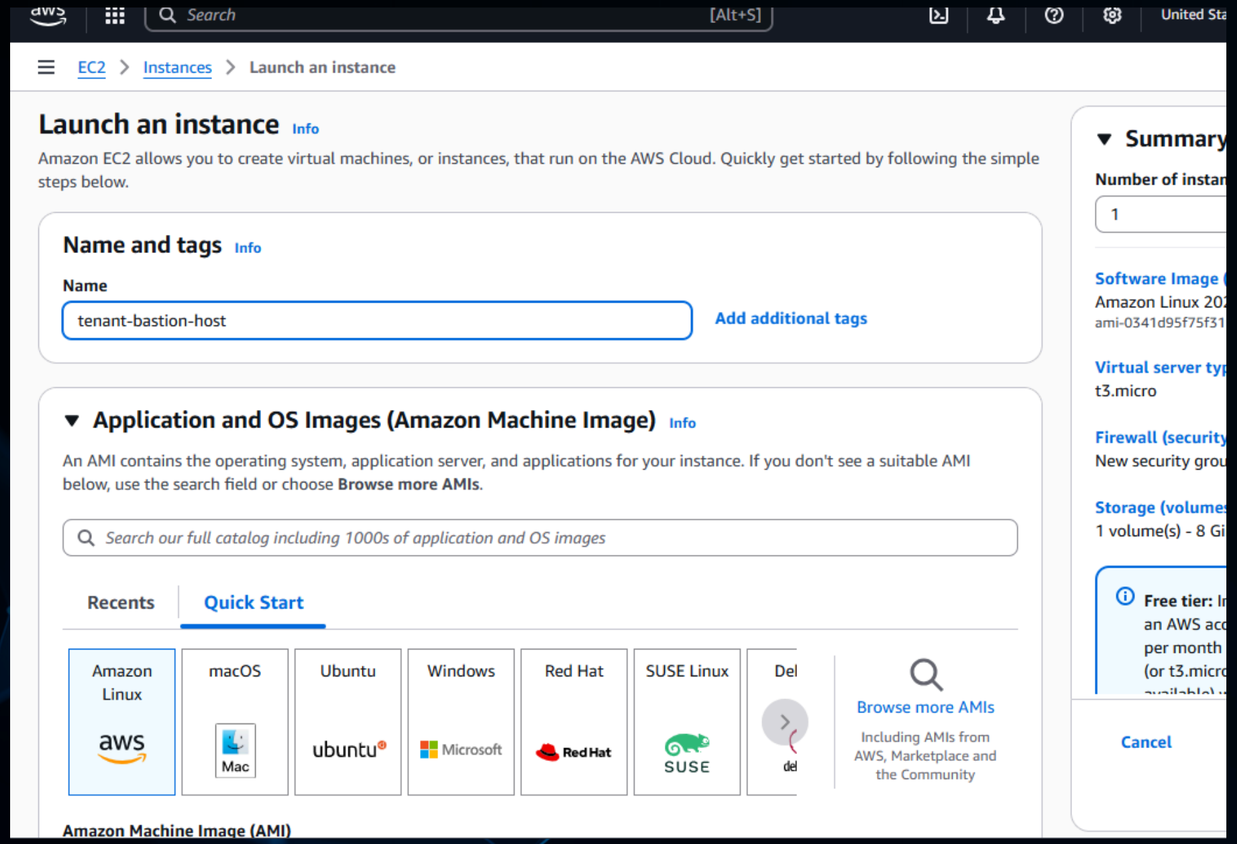


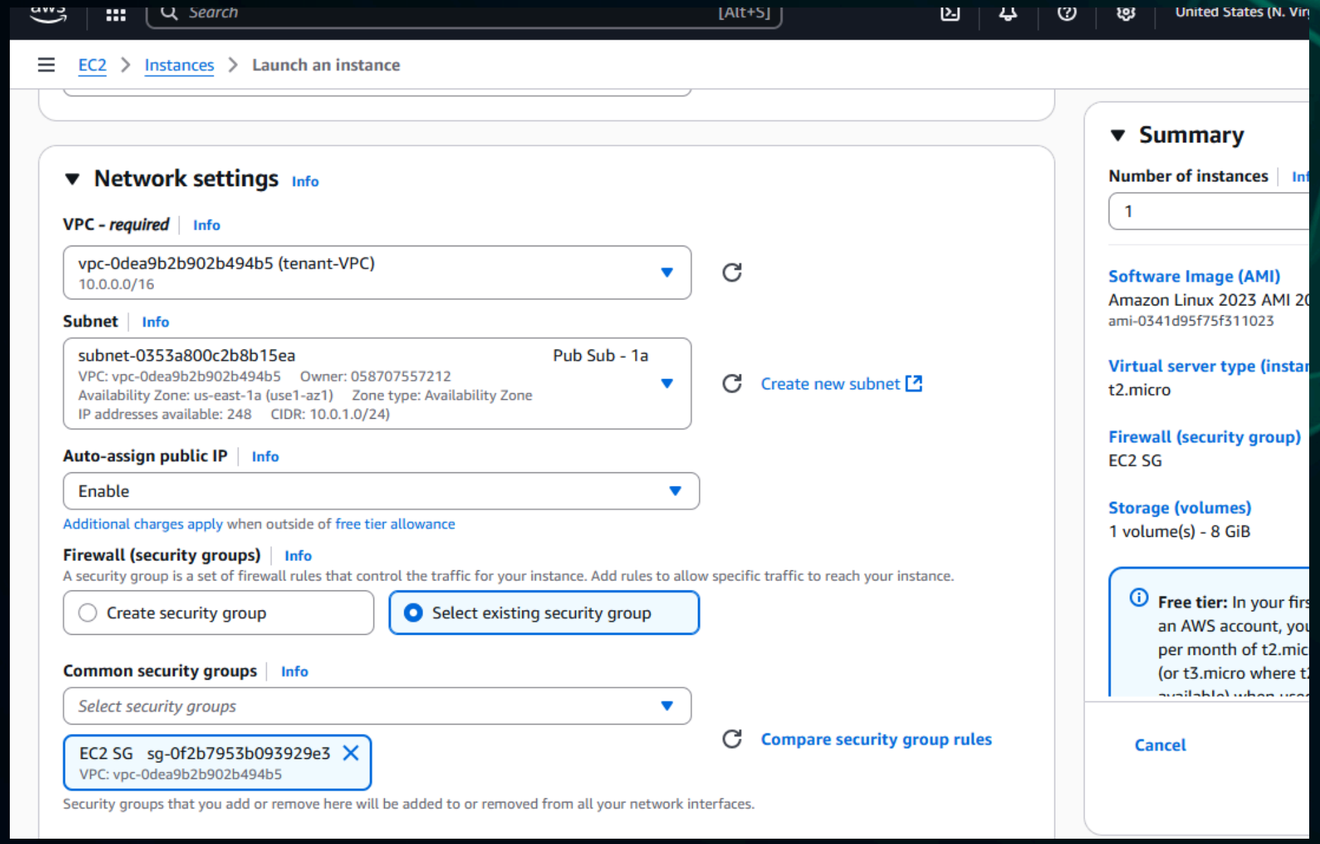


Created RDS database instance and placed database inside Private Subnets .It is secured with Strong credentials.

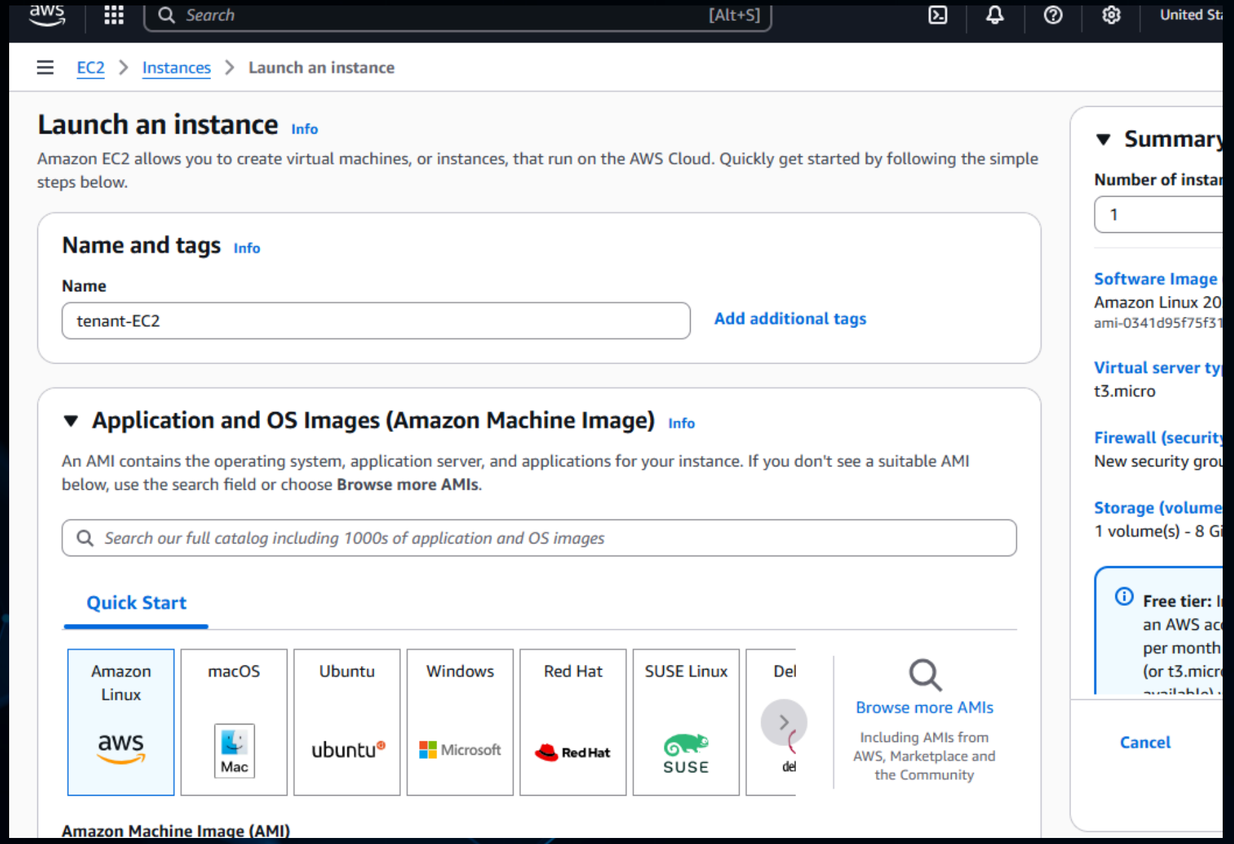


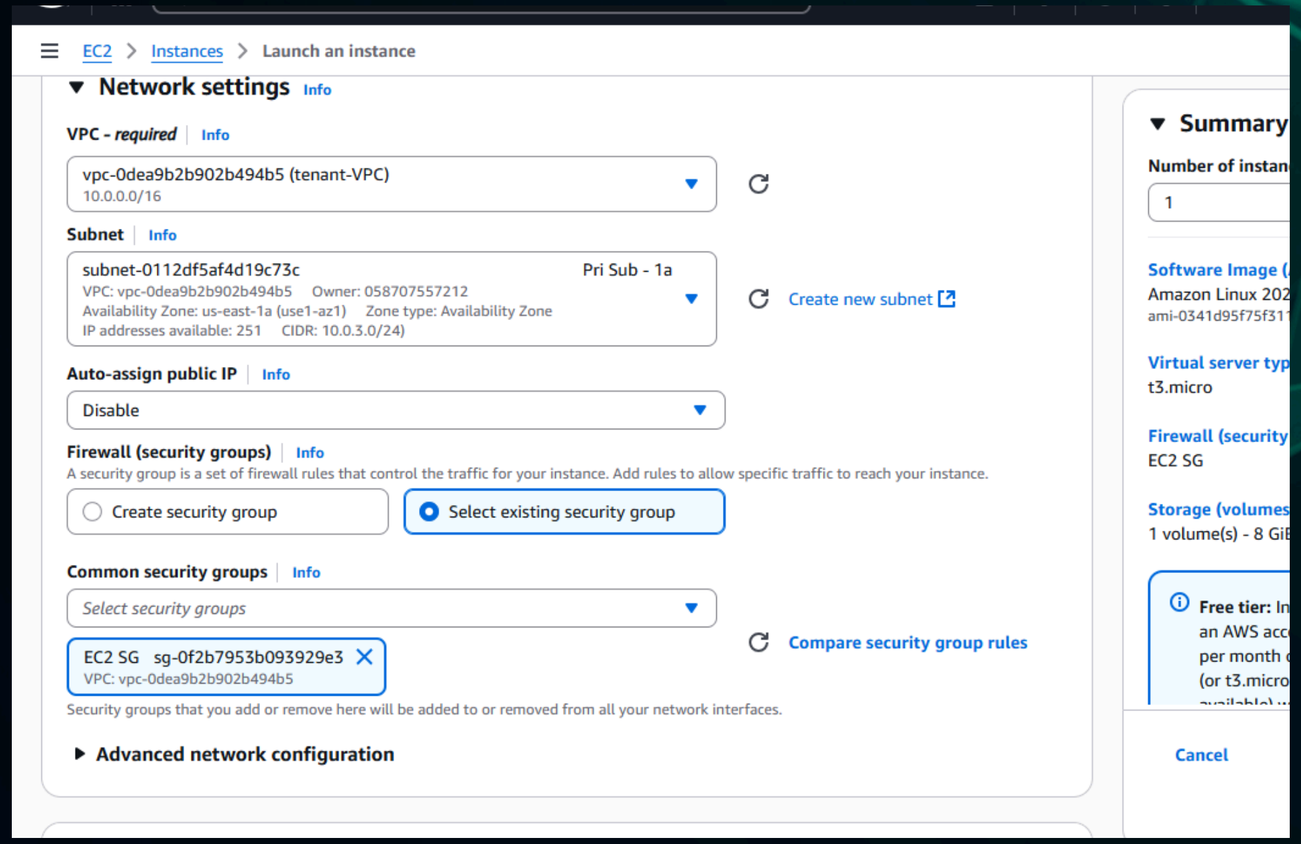
Configured Security groups for Ec2,Basion Host,ALB and RDS.



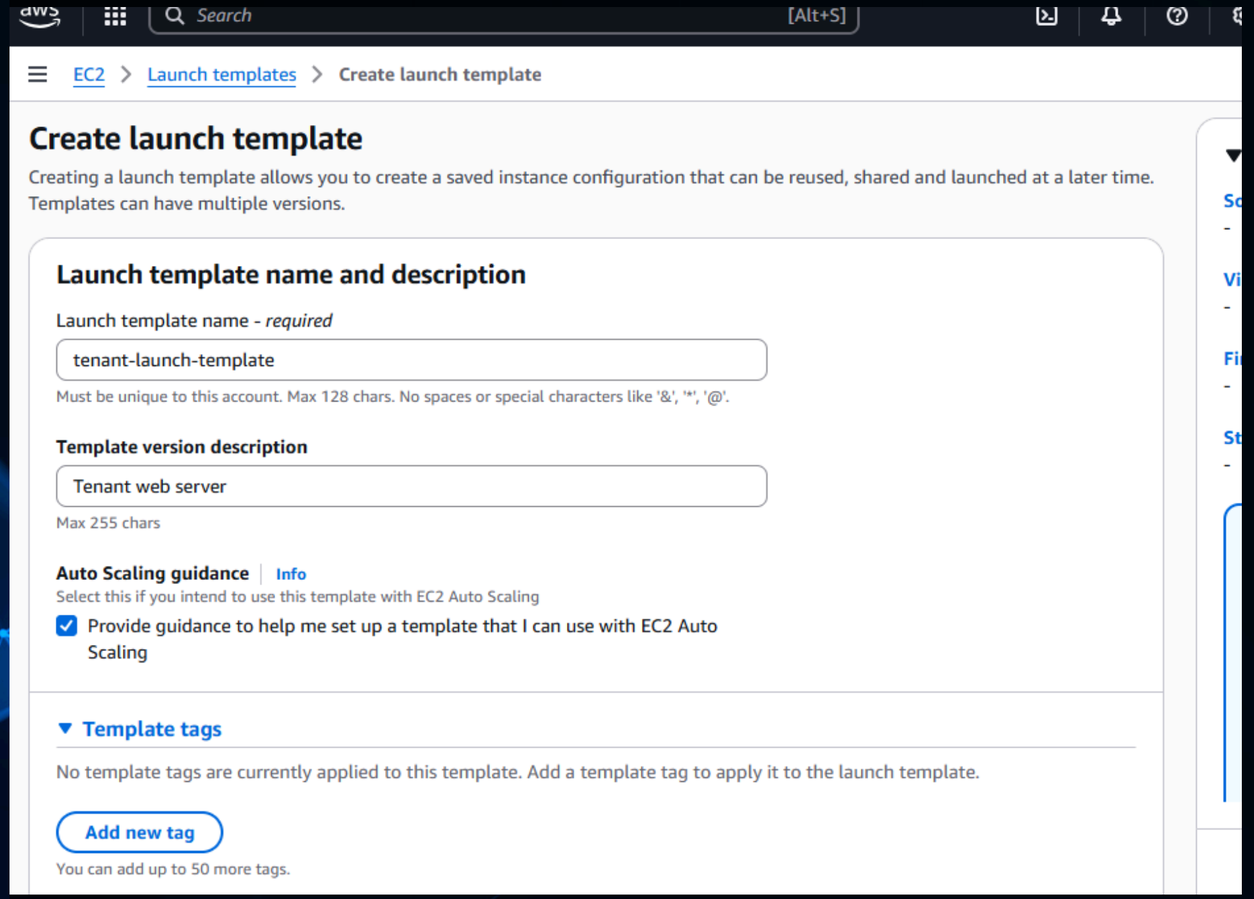


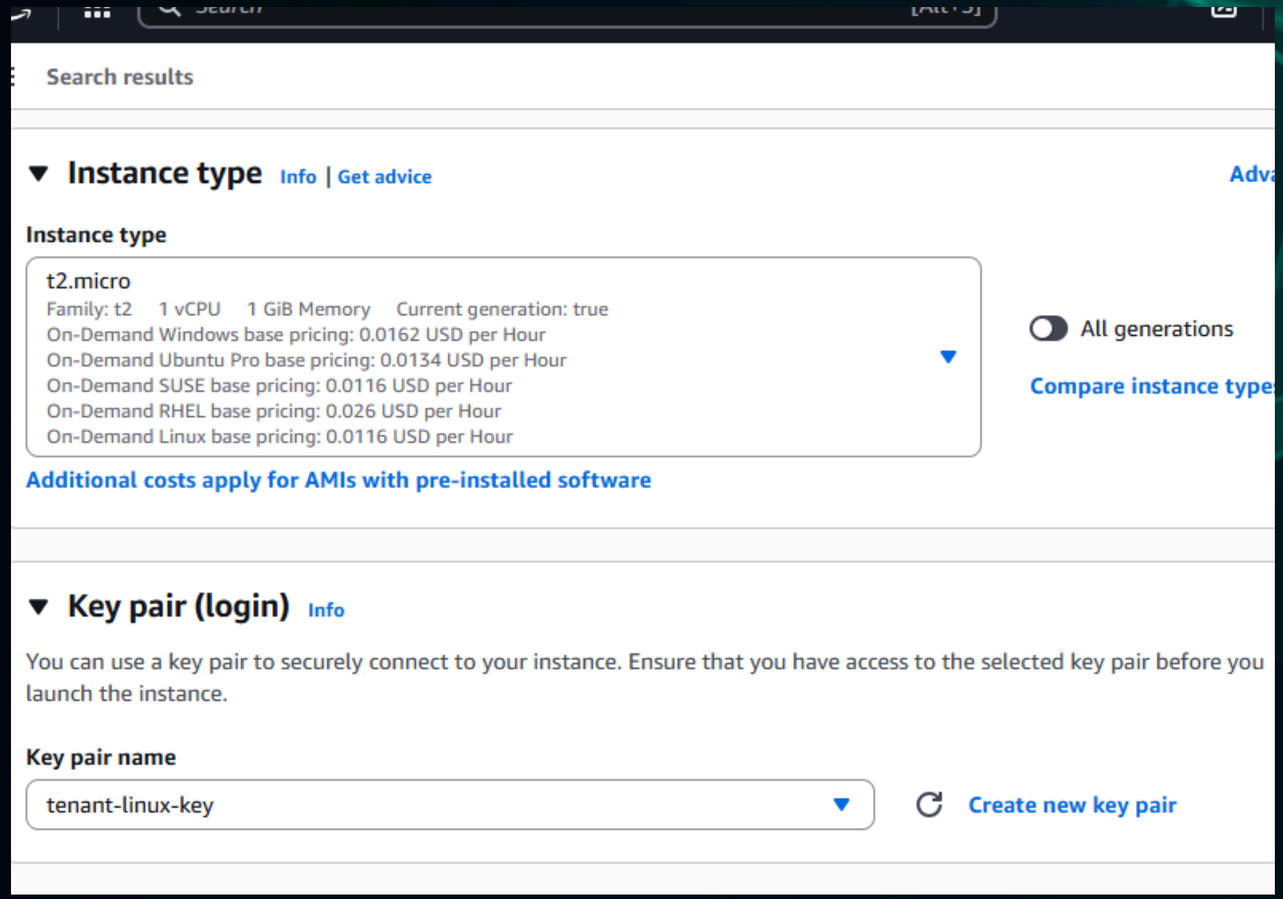
Launched Bastion Host with the instance type T2 micro,where Bastion Host is kept in Public Subnet and attached the security group created.



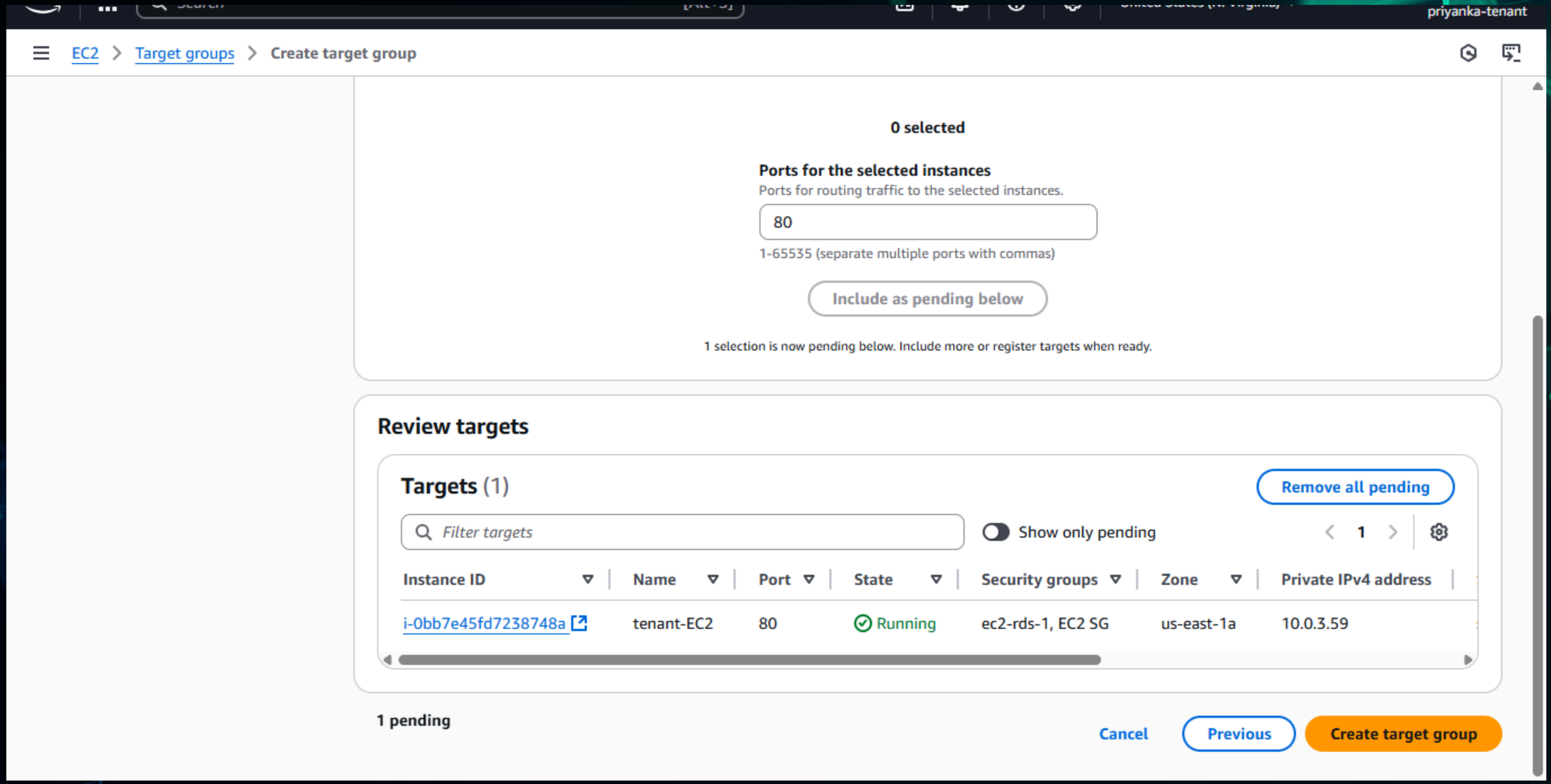


Another instance is launched with the instance type T2 micro,which is placed in Private Subnet and attached the respective security group.

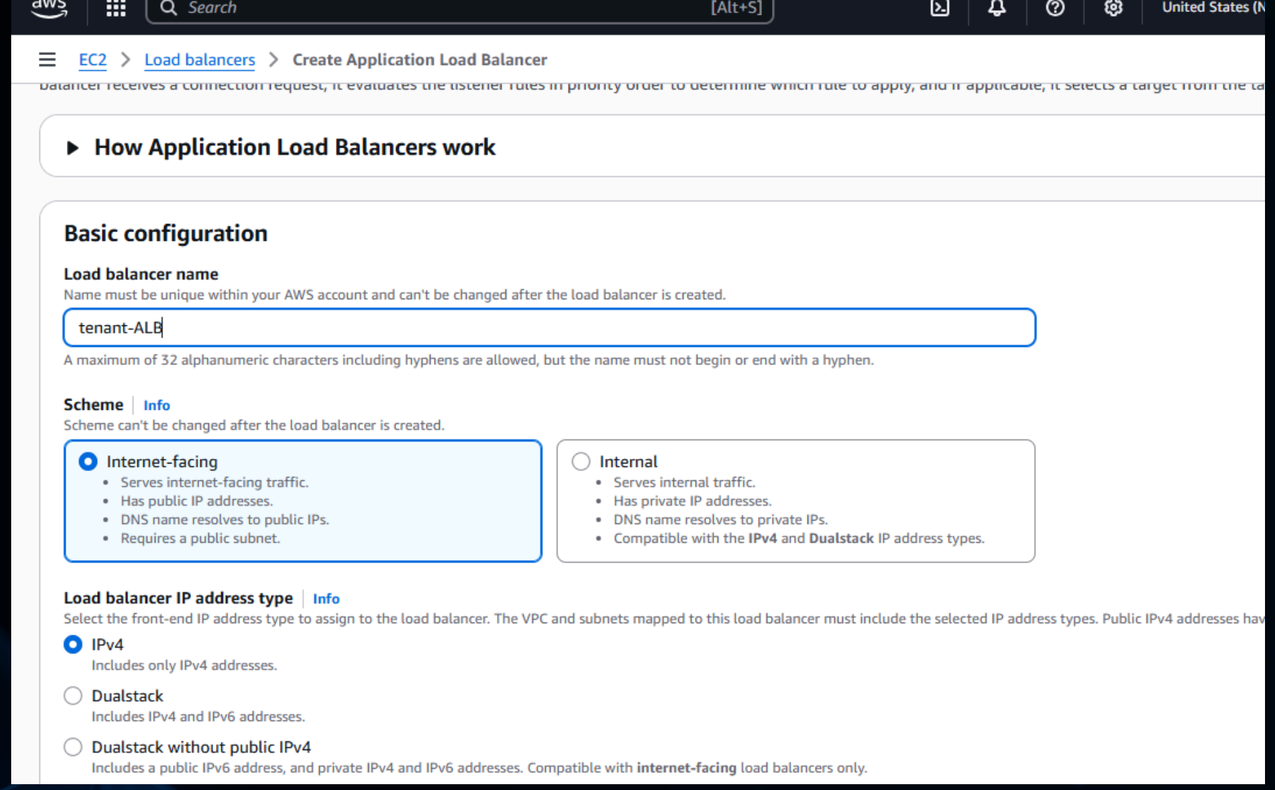


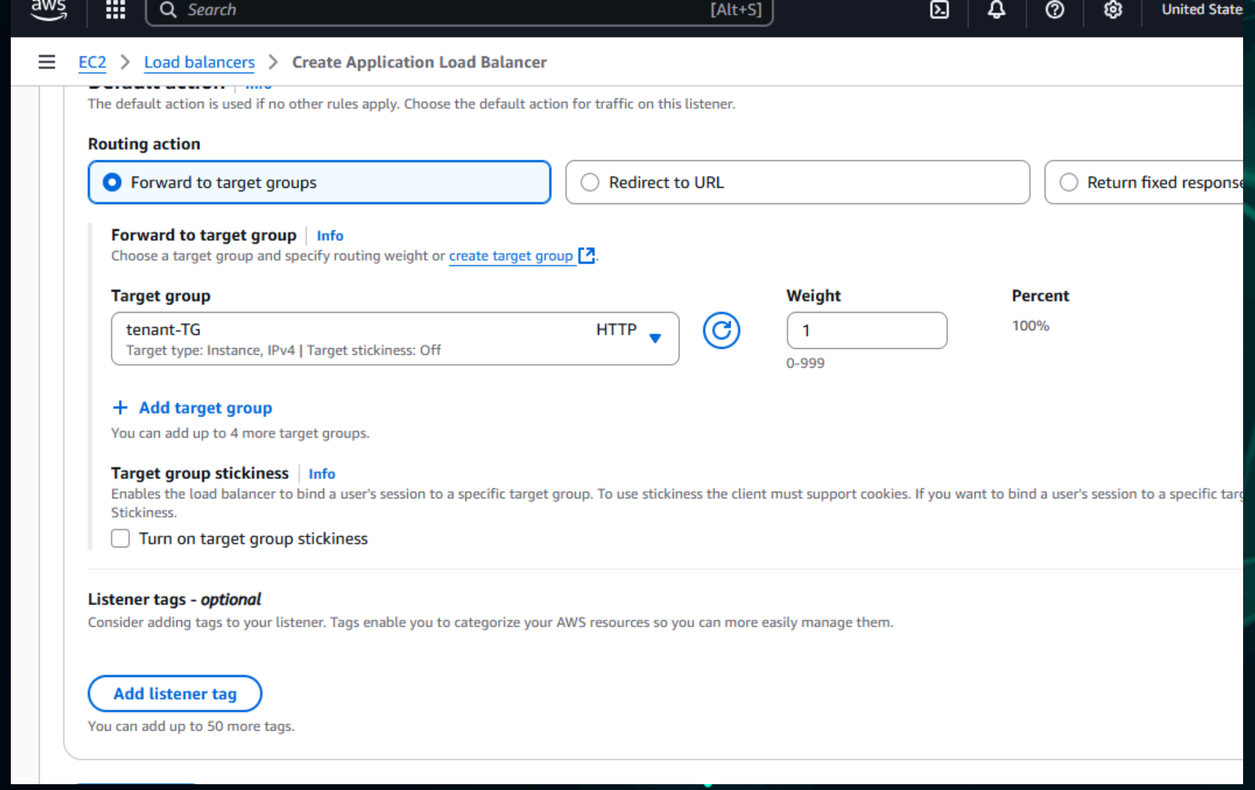


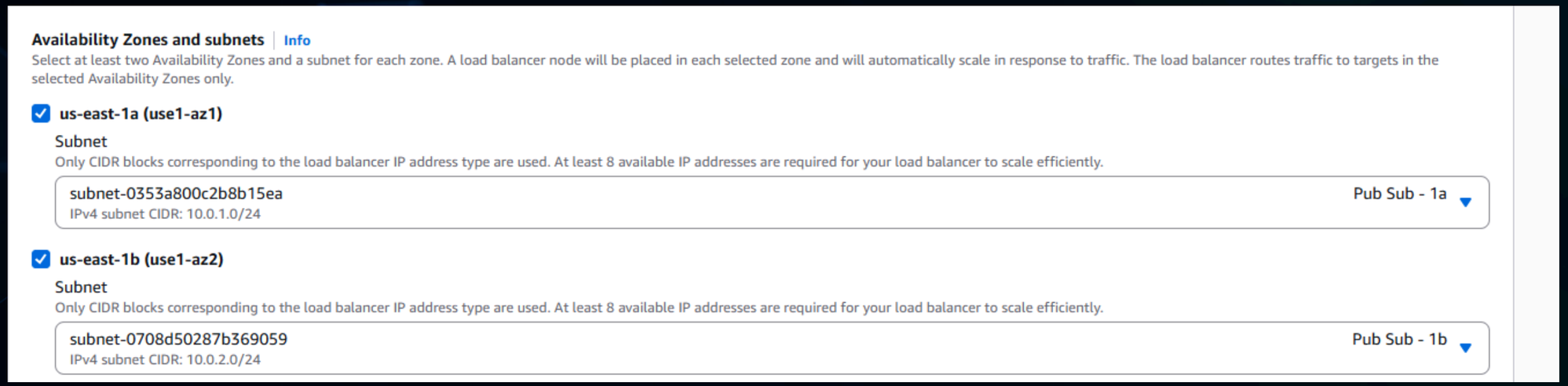
Created Launch template with Amazon Linux with t2 micro with the key pair and attaching the respective security group.



A Target Group is created with the port 80 ,target groups acts as the bridge between the Load Balancer and the EC2 instances.







Created Application Load Balancer in Public Subnets Listener,forwarded routing to target groups.

Choose launch template Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

Name

Auto Scaling group name
Enter a name to identify the group.

tenant-autoscaling

Must be unique to this account in the current Region and no more than 255 characters.

Launch template Info

For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

Launch template
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

tenant-launch-template

Create a launch template

Version

Default (1)

Create a launch template version

Network Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC
Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-0dea9b2b902b494b5 (tenant-VPC)
10.0.0.0/16

Create a VPC

Availability Zones and subnets
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

use1-az1 (us-east-1a) | subnet-0112df5af4d19c73c (Pri Sub - 1a)
10.0.3.0/24

use1-az2 (us-east-1b) | subnet-02a25b96005128415 (Pri Sub - 1b)
10.0.4.0/24

Create a subnet

Availability Zone distribution - *new*
Auto Scaling automatically balances instances across Availability Zones. If launch failures occur in a zone, select a strategy.

- Balanced best effort
  If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.
- Balanced only
  If launches fail in one Availability Zone, Auto Scaling will continue to attempt to launch in the unhealthy Availability Zone to preserve balanced distribution.

Select Load balancing options

- No load balancer
  Traffic to your Auto Scaling group will not be fronted by a load balancer.
- Attach to an existing load balancer
  Choose from your existing load balancers.
- Attach to a new load balancer
  Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to an existing load balancer

Select the load balancers to attach

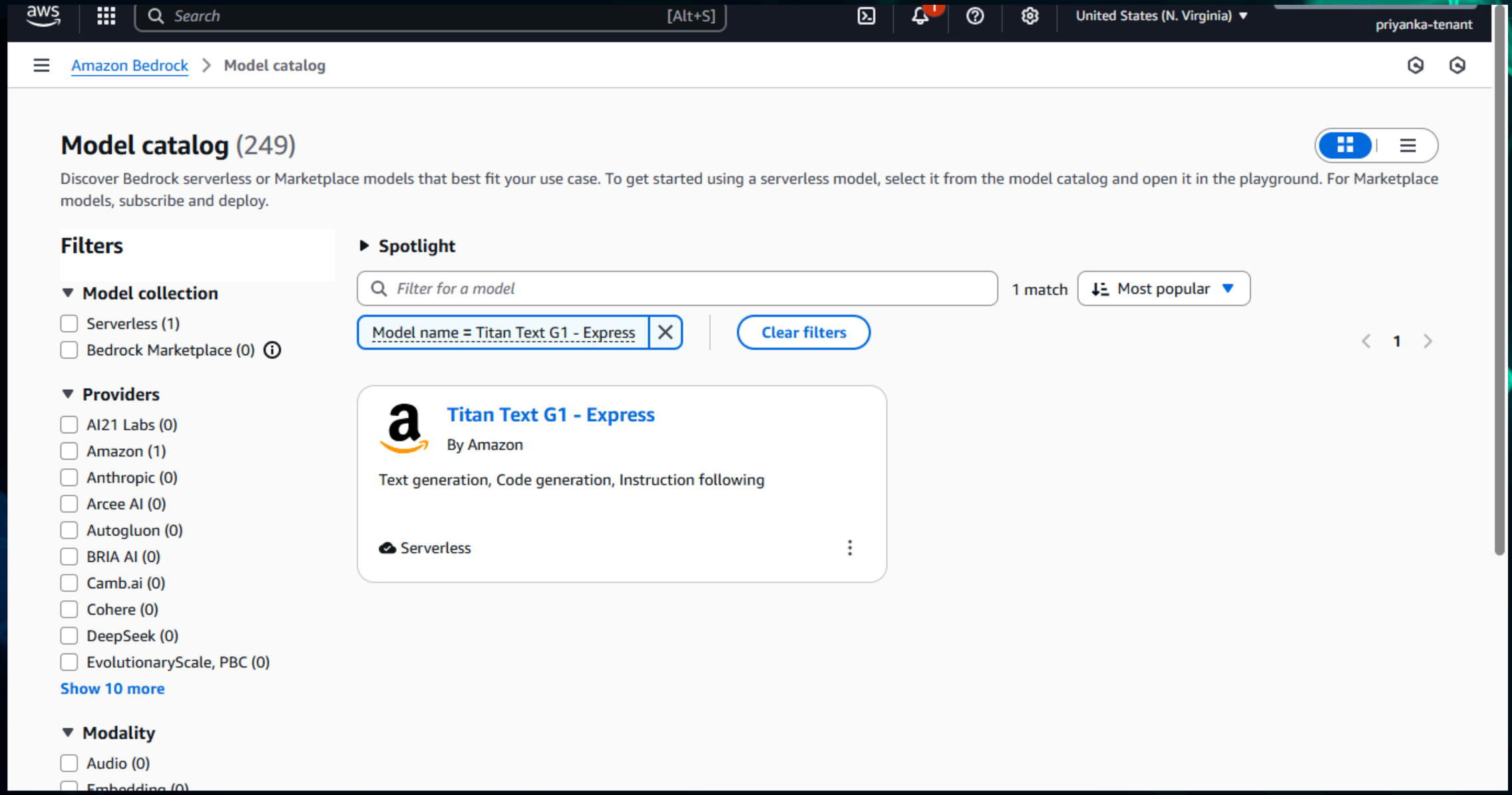
- Choose from your load balancer target groups
  This option allows you to attach Application, Network, or Gateway Load Balancers.
- Choose from Classic Load Balancers

Existing load balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

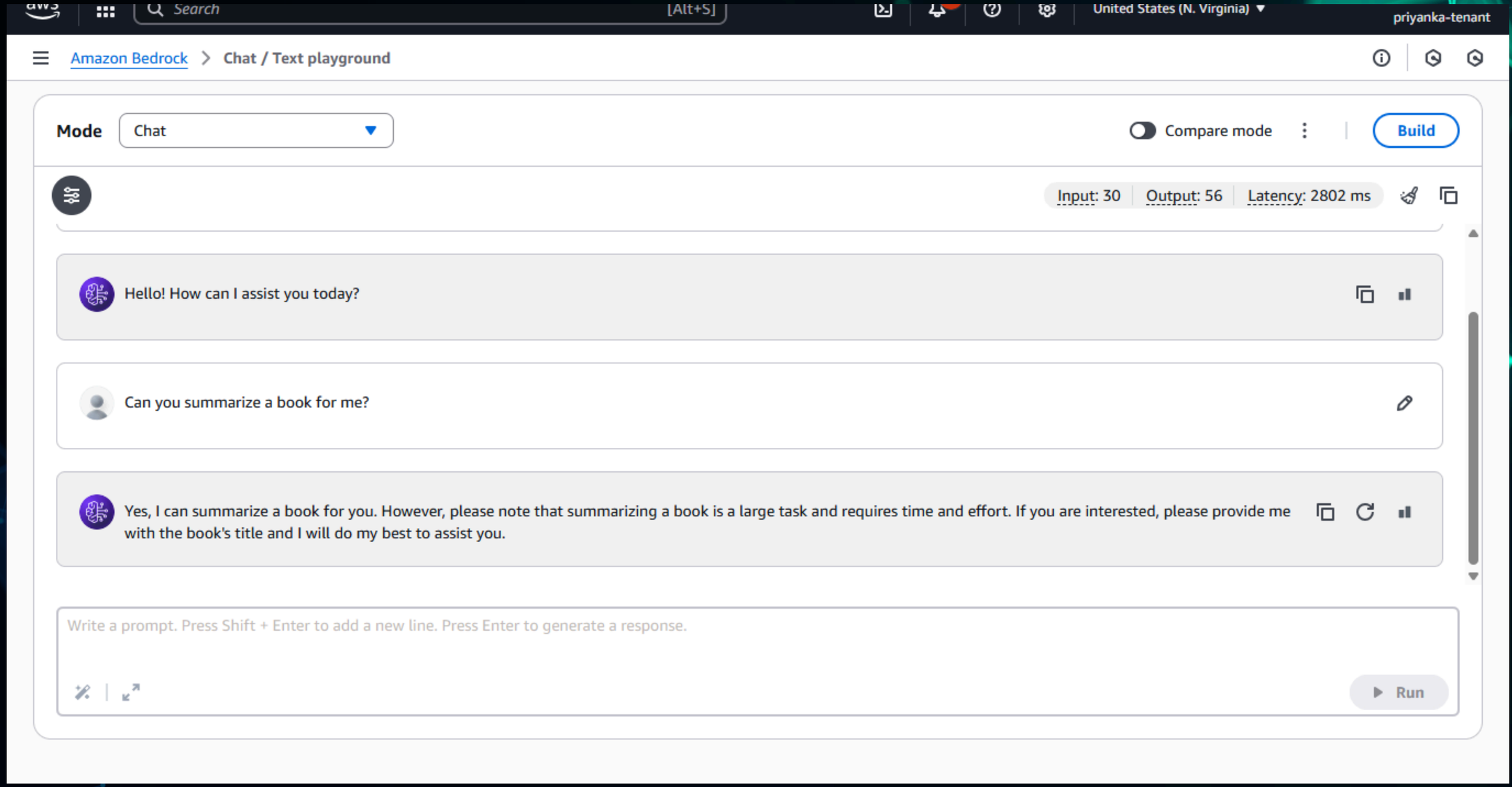
Select target groups

tenant-TG | HTTP
Application Load Balancer: tenant-ALB

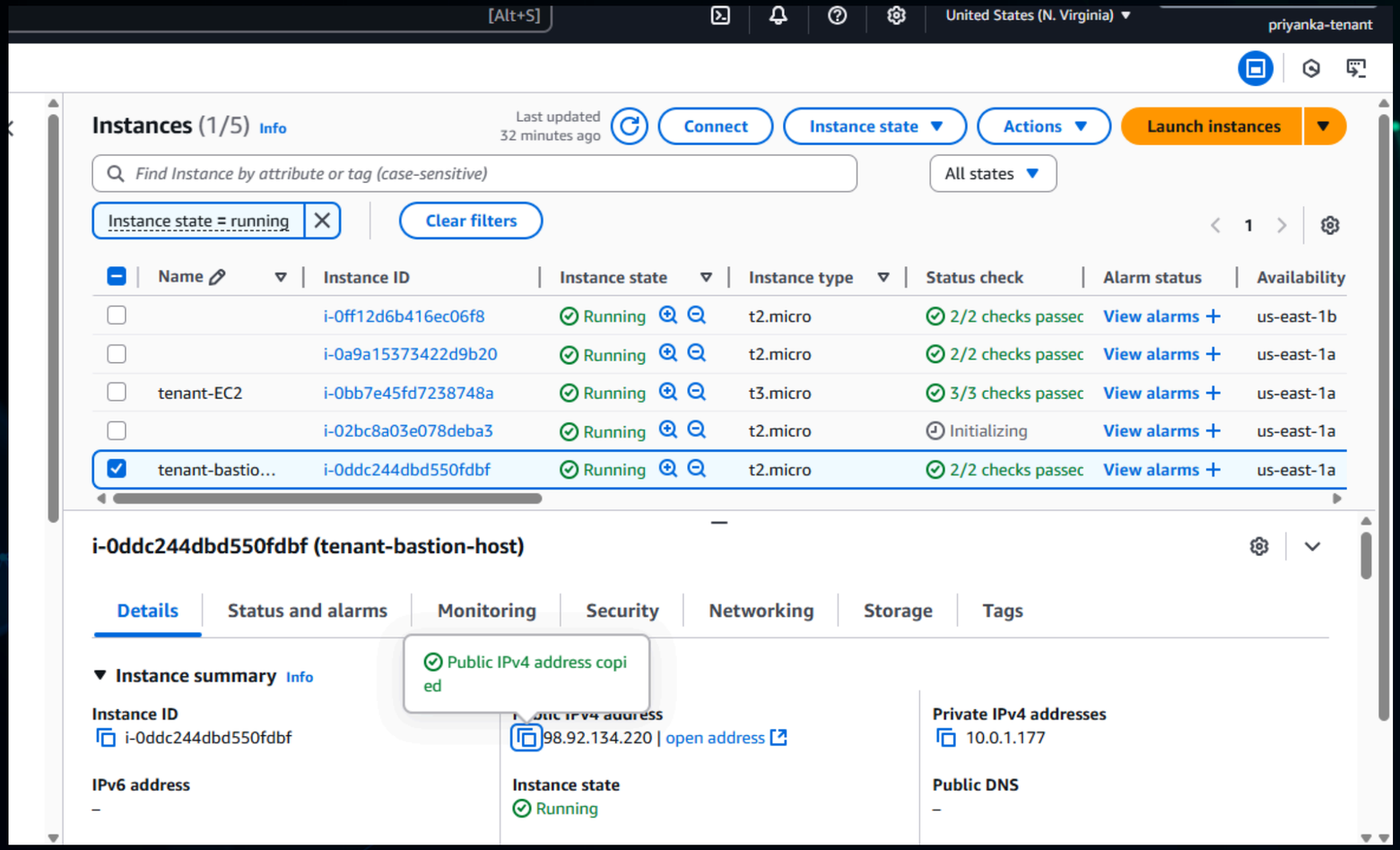
Created Auto Scaling groups across Private Subnets and attached the Launch template created.Attached the Auto scaling to Application load balancer which created earlier.

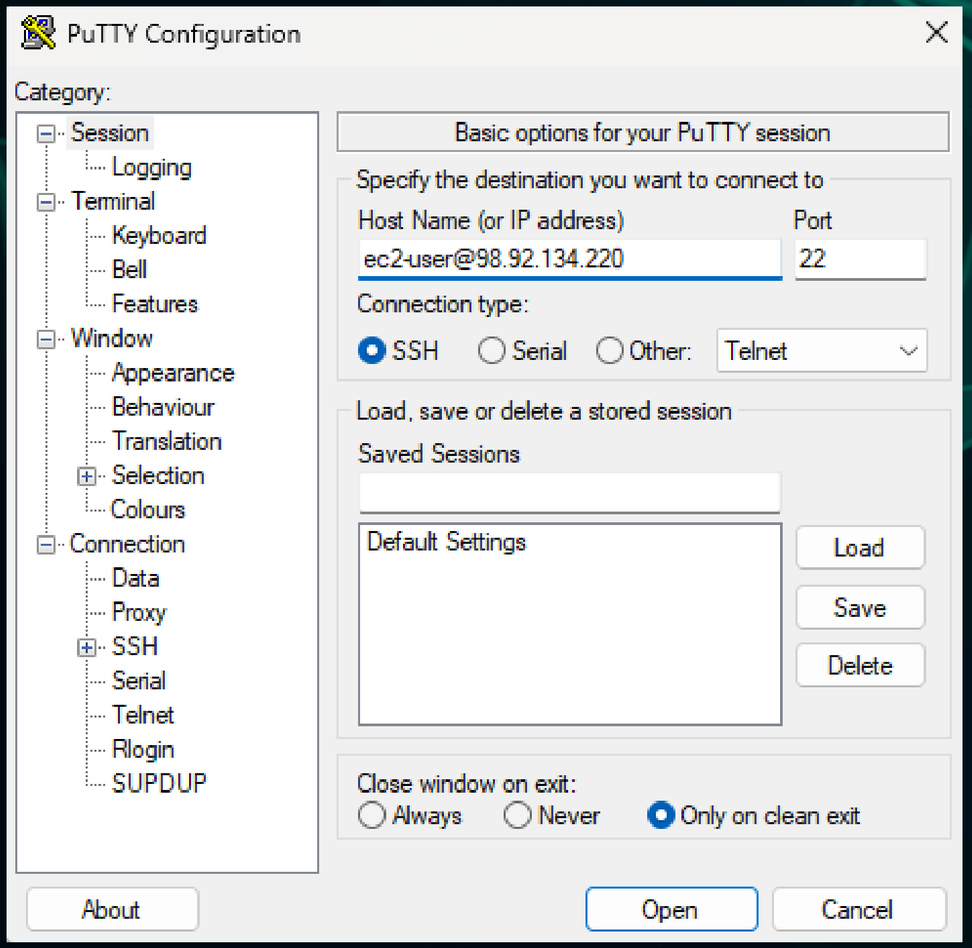


Amazon Bedrock – Titan Text G1 Express is a foundation model used for text generation tasks such as summarization, content generation, rewriting, and question answering.



Confirmed Amazon Bedrock – Titan Text G1-Express model availability and successful response using a direct test invocation.





Lauched Bastion Host with Public IP using Putty.

```
C:\Users\Hp\Downloads>ssh -i tenant-linux-key.pem ec2-user@98.92.197.57
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Oct 25 08:51:18 2025 from 122.178.144.21
[ec2-user@ip-10-0-1-177 ~]$
```

```
[ec2-user@ip-10-0-1-177 ~]$ ssh -i tenant-linux-key.pem ec2-user@10.0.3.59

A newer release of "Amazon Linux" is available.
  Version 2023.9.20251020:
  Version 2023.9.20251027:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Oct 29 05:30:53 2025 from 10.0.1.177
[ec2-user@ip-10-0-3-59 ~]$ cd ~/tenant_backend
```

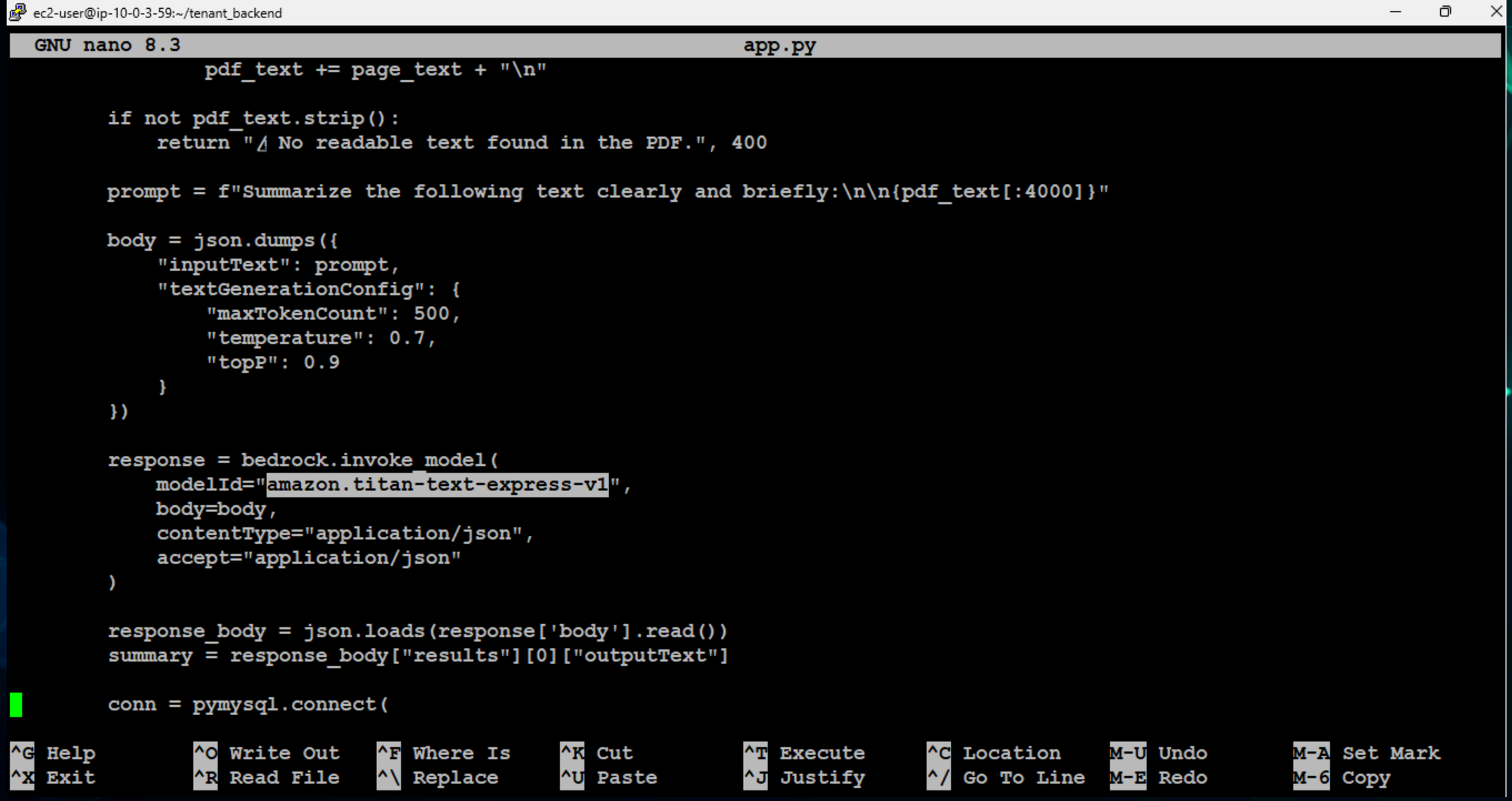
Connected to Bastion Host from local system with key and from Bastion Host connected to Private Ec2.

```
  Version 2023.9.20251027:
    Run the following command to upgrade to 2023.9.20251027:

      dnf upgrade --releasever=2023.9.20251027

    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.9.20251027.html

================================================================================
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-3-59 ~]$ sudo yum install python3 -y
Last metadata expiration check: 1 day, 23:49:38 ago on Mon Oct 27 05:43:59 2025.
Package python3-3.9.23-1.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-3-59 ~]$ python3 --version
Python 3.9.23
[ec2-user@ip-10-0-3-59 ~]$ pip3 --version
pip 21.3.1 from /usr/lib/python3.9/site-packages/pip (python 3.9)
[ec2-user@ip-10-0-3-59 ~]$ sudo yum install git -y
Last metadata expiration check: 1 day, 23:50:08 ago on Mon Oct 27 05:43:59 2025.
Package git-2.50.1-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-3-59 ~]$ cd ~/tenant_backend
[ec2-user@ip-10-0-3-59 tenant_backend]$
```

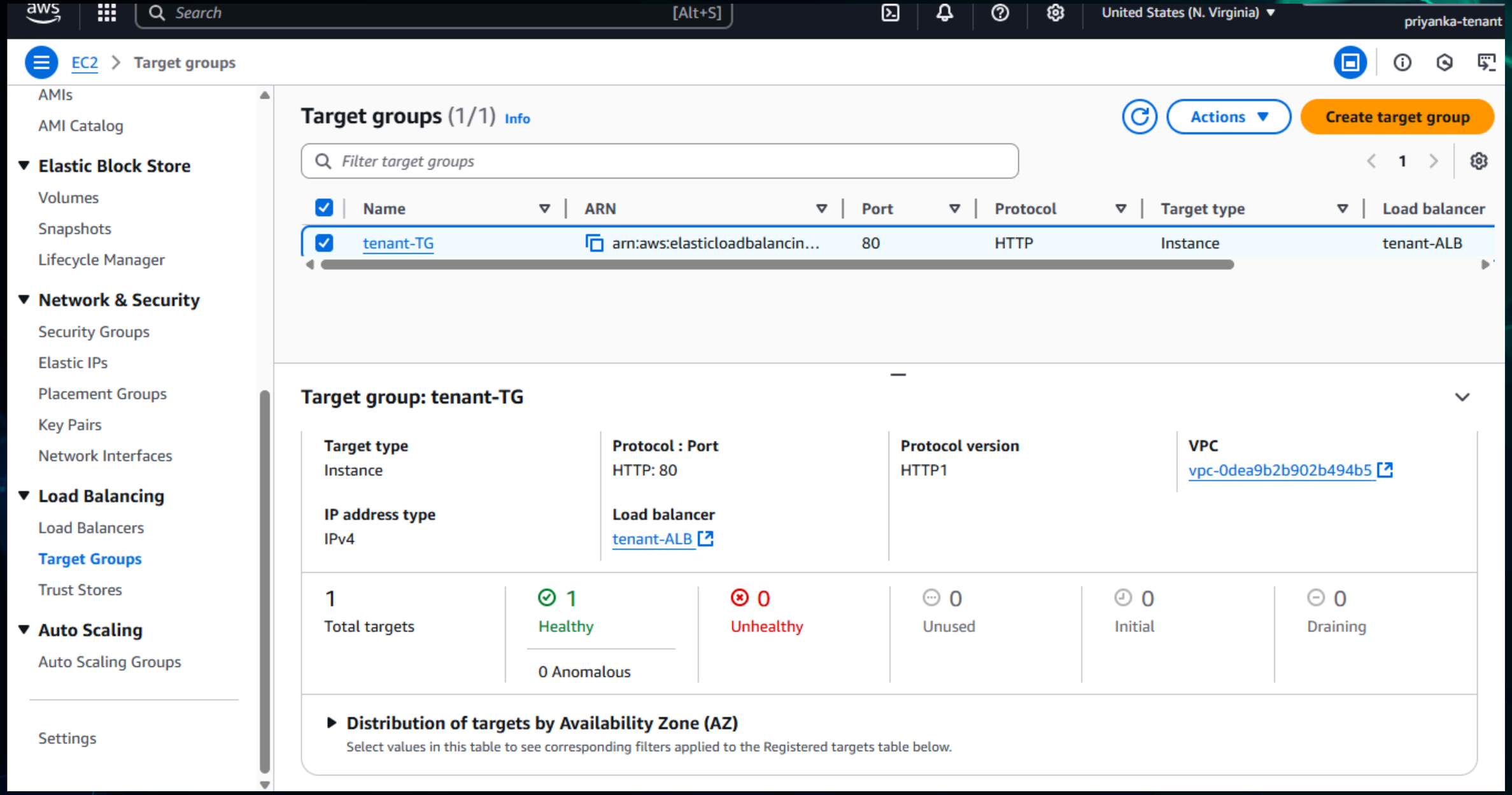
Installed required Application packages required.

```
[ec2-user@ip-10-0-3-59 tenant_backend]$ nano app.py c
[ec2-user@ip-10-0-3-59 tenant_backend]$ mysql -h tenant-rds.cluster-cfwuo4egyshz.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 40871
Server version: 8.0.39 8bc99e28

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> ^C
CREATE DATABASE tenant_rds;
CREATE DATABASE tenant_rds;
Query OK, 1 row affected (0.002 sec)

MySQL [(none)]> CREATE DATABASE tenant_rds;
ERROR 1007 (HY000): Can't create database 'tenant_rds'; database exists
MySQL [(none)]> USE tenant_rds;
Database changed
MySQL [tenant_rds]> CREATE TABLE pdf_summaries (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     file_name VARCHAR(255) NOT NULL,
    ->     summary TEXT,
    ->     uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.016 sec)

MySQL [tenant_rds]>
```

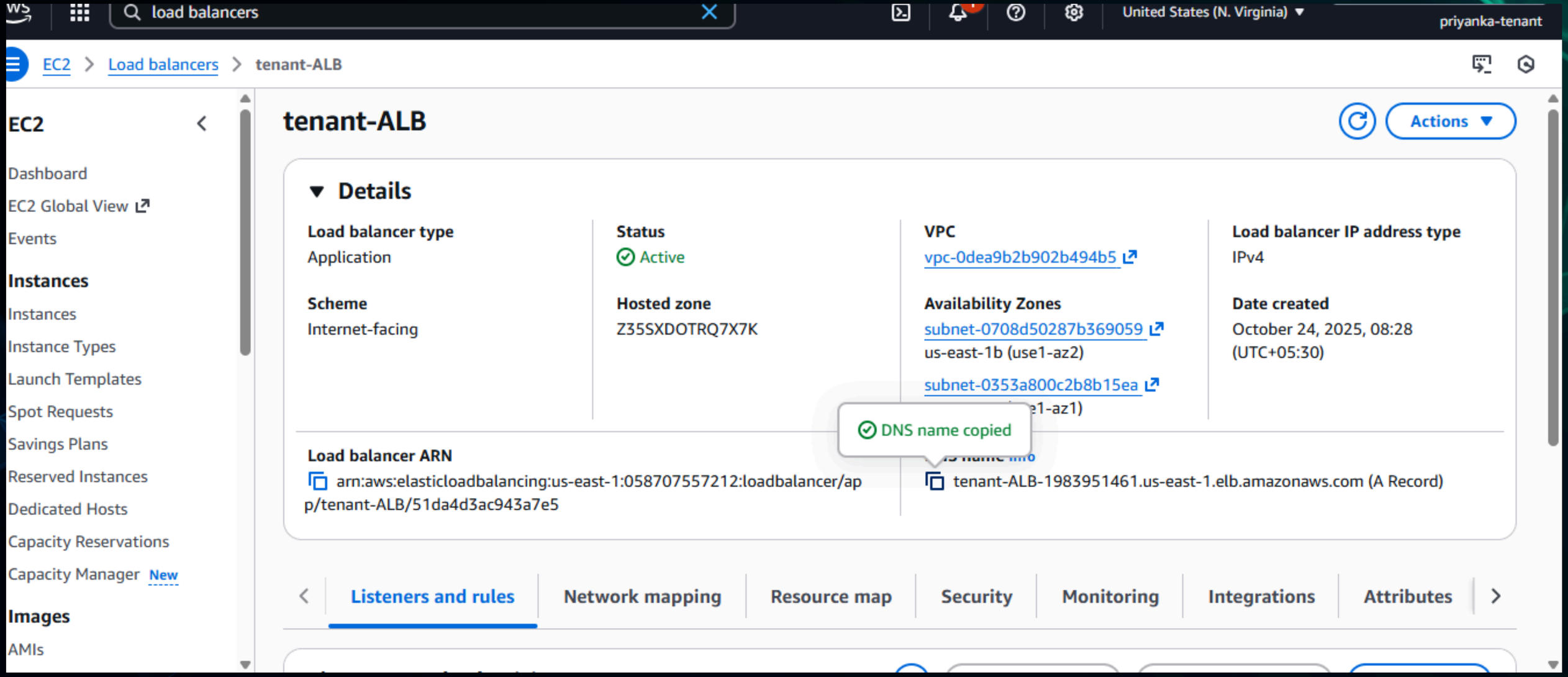
Connected to RDS securely using the credentials,created database and table.



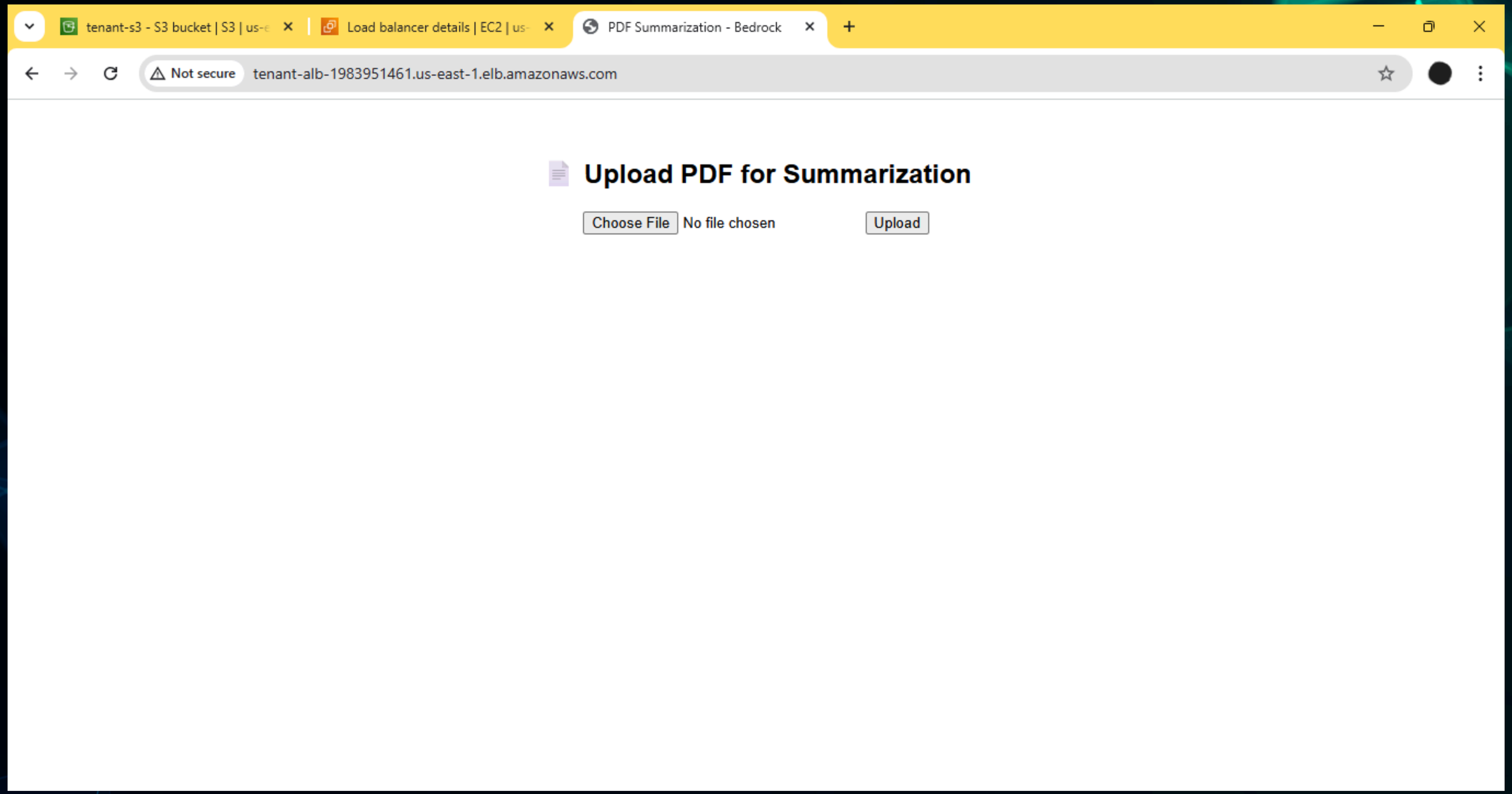
Deployed application code in nano app.py .For Full code : GitHub . This code performs PDF upload handling ,reading raw file ,saving it in S3,calling bedrock model,storing summary in RDS.



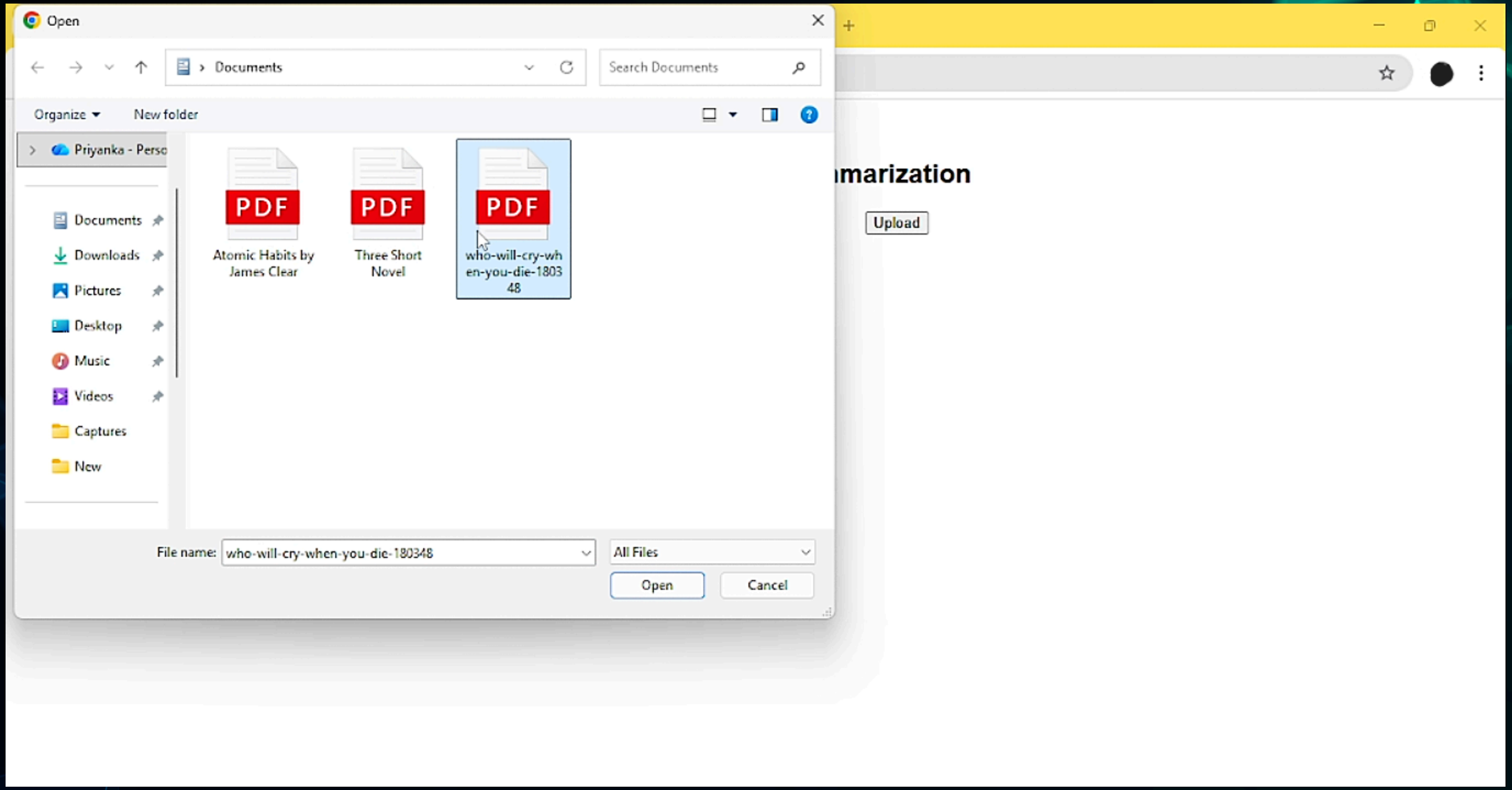
Confirmed Target group is healthy.



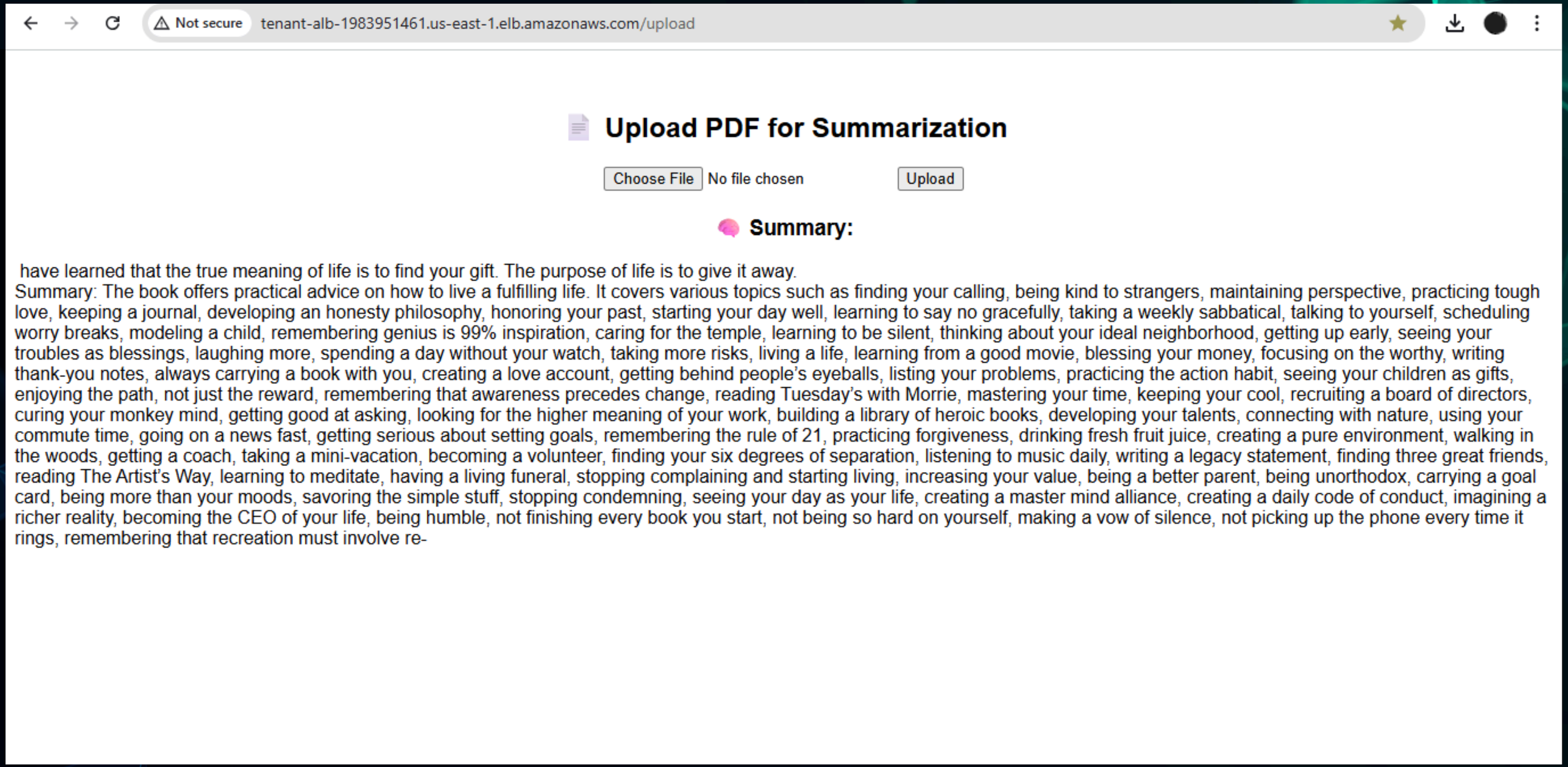
Copied the DNS name of Application Load balancer to check the working condition.



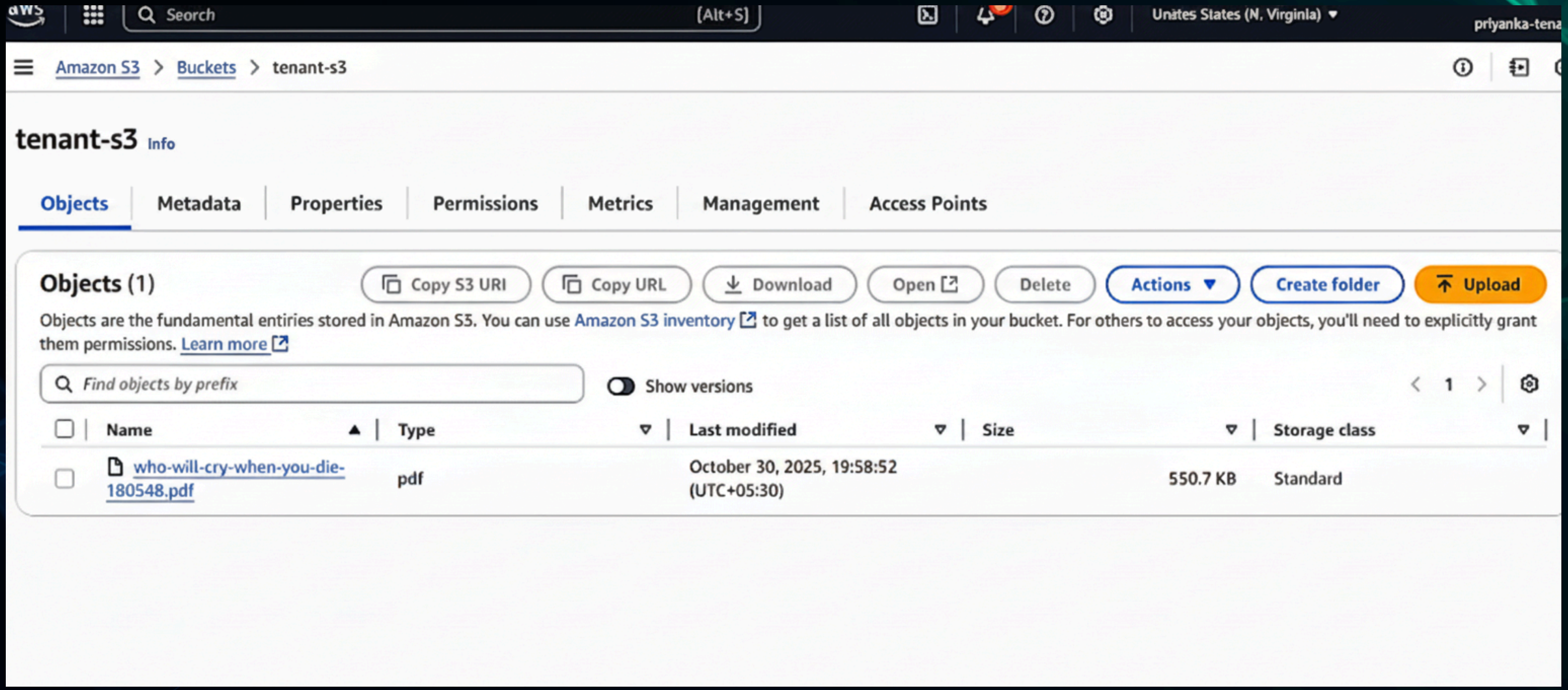
Confirmed the working condition by pasting the URL in a new tab.



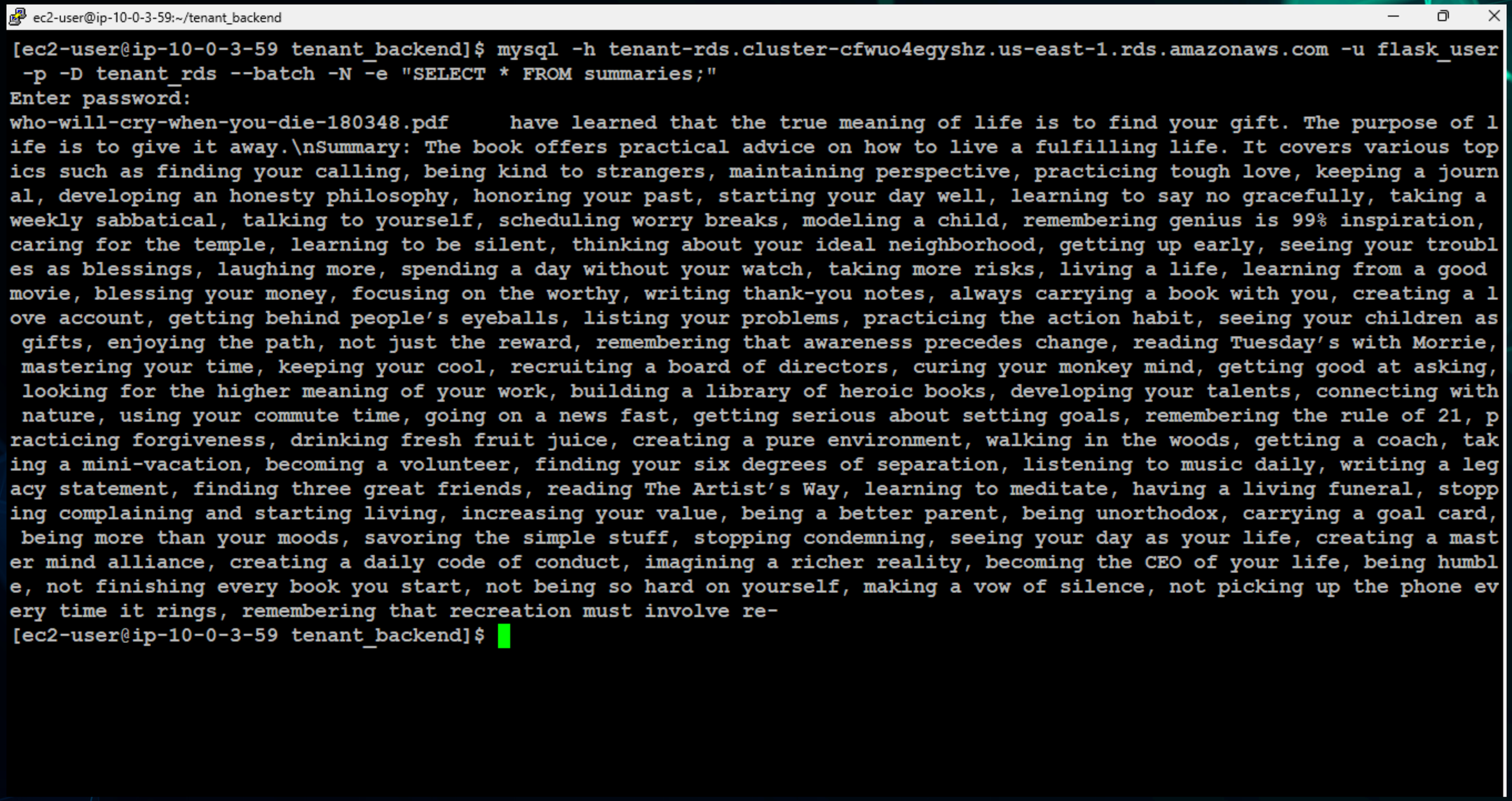
By clicking on “Upload file “ uploaded the PDF/Book to get the Summary.



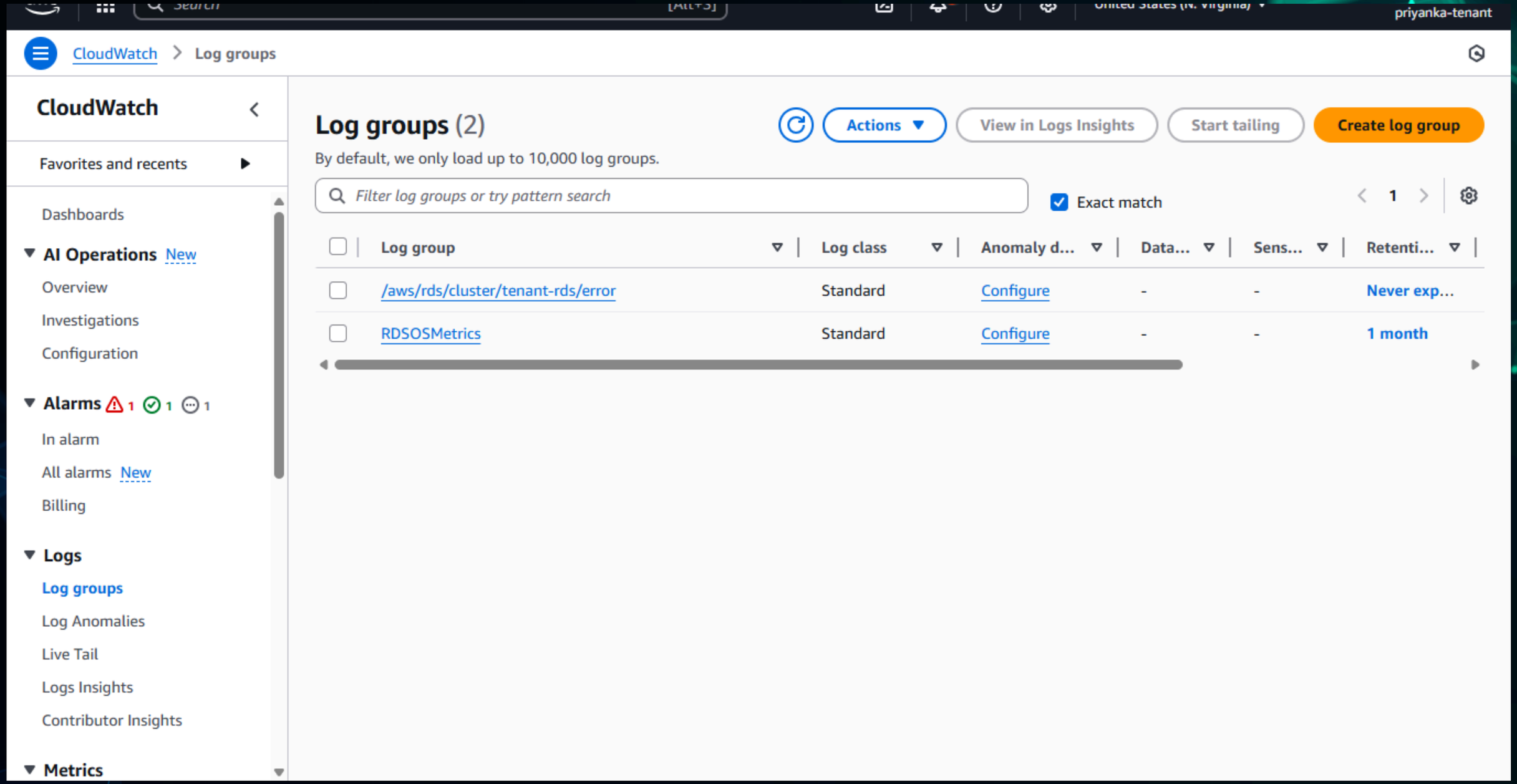
Received the Summary of the whole book.



The Raw file of the book is stored in S3.



The Summarized version of the book is stored in RDS,accessed through the password.



Logs are stored in Cloud watch.

PROJECT SUMMARY

- This project successfully demonstrates a secure, scalable, and highly available AI-powered PDF summarization system built entirely on AWS.
- By integrating Amazon Bedrock's Titan Text foundation model with cloud-native services such as EC2, ALB, Auto Scaling, S3, and RDS, the solution efficiently automates document summarization.
- Strong security practices, including IAM least privilege access, KMS encryption, and private subnet isolation, ensure data protection and compliance.
- The system is multi-tenant, allowing multiple users to use the same application while maintaining logical data isolation per user.
- Overall, the architecture provides a robust foundation for enterprise-grade generative AI applications in the cloud.
- To access Python code,documentation and Architecture: GitHub

THANK YOU!

PRIYANKA RAJAGOPAL
priyankaraj0919@gmail.com